Czech Technical University in Prague
Faculty of electrotechnics

Master's thesis

Virtual Center for Educators of Seniors

Jaroslav Kortus

Supervisor: Ing. Božena Mannová, M.Math.

Study program: Electrotechnics and informatics

Branch: Informatics and computer science

2007

Task

Prohlášení

Prohlašuji, že jsem svoji diplomovou práci vypracoval samostatně a použil jsem výhradně podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne .....

Acknowledgments

I would like to thank to my supervisor Ing. Božena Mannová, M.Math. for the proposition of the topic, for the help with formal aspects of the thesis and for guidance through the whole process of writing this thesis.

Special thank goes to my friend Ing. Helena Holíková for a proof-reading.

Dedicated to
my parents Jaroslav and Božena
with gratitude for their
endless patience, boundless support and love.

Summary

This final project deals with the problems addressed by development of enterprise server-side web applications in Java programming language. I was assigned to study a plenty of materials, which are closely related to mentioned branch and hand it over in order to get the reader acquainted with the problems in an appropriate way.

I was expected to apply and demonstrate such theoretical knowledge, implementing of one specific web application for support teaching of computer's acquirements, the Virtual Center for Educators of Seniors.

Anotace

Tato diplomová práce se zabývá problematikou tvorby podnikových serverových webových aplikací v jazyce Java. Cílem bylo nastudovat materiály, které s danou problematikou úzce souvisí a podat je tak, abych čtenáře vhodnou formou uvedl do problematiky.

Nasbírané teoretické znalosti jsem poté měl aplikovat a demonstrovat tím jejich využití při implementaci specifické webové aplikace pro podporu výuky počítačových dovedností, Virtuálního centra pro vzdělavatele seniorů.

Keywords

Java™ EE 5, Java EE Application Model, SJSAS 9 (Glassfish), Java Servlet, JSP, JavaBeans, JSF API, MVC, JAAS, JTA, UI Components, Java Persistence API, Security, NetBeans IDE, MySQL, Virtual Center

Contents

List of Figures **xviii**

List of Tables **xix**

I Theoretical Background – Java EE 5 **3**

1 Java EE overview **5**
1.1 Enterprise Application . . . 6
1.2 Application Architecture . . . 6
1.2.1 Model View Controller . . . 7
1.2.1.1 Model1 . . . 7
1.2.1.2 Model2 . . . 8
1.3 Java EE Components . . . 8
1.4 Java EE Containers . . . 9
1.5 Java EE APIs . . . 10

2 Java Servlets **11**
2.1 Servlet vs. CGI . . . 11
2.2 Servlet's life–cycle . . . 12
2.2.1 Writing `init()` method . . . 13
2.2.2 Writing `do`*`Something`*`()` Method . . . 13
2.2.3 Writing servlet's `destroy()` method . . . 13
2.3 Filtering Request and Responses . . . 13
2.3.1 Creating of filter classes . . . 14
2.4 Constructing of responses . . . 14
2.4.1 Including other resource . . . 16
2.4.2 Transferring Control to Another Component . . . 16
2.5 Session management . . . 16
2.6 Sharing information . . . 16
2.7 New features in Servlet 2.5 . . . 16
2.8 Why Servlets overrun CGI . . . 17

3 JavaServer Pages Technology **19**
3.1 JSP's Life-cycle . . . 19
3.2 Directives . . . 19
3.3 Scripting elements . . . 21
3.3.1 Expressions . . . 21
3.3.2 Scriptlets . . . 21
3.3.3 Declarations . . . 21
3.4 Commentaries . . . 21

3.5 Tags . . . 21
3.6 Sharing objects . . . 22
3.7 JavaBeans Components . . . 22
3.8 Unified Expression Language . . . 22
3.9 JSP Documents . . . 23
3.10 JavaServer Pages Standard Tag Library . . . 23

4 JavaServer Faces **25**
4.1 What is JavaServer Faces? . . . 25
4.2 The MVC architecture of JSF . . . 25
4.3 The Life-cycle of a JSF Page . . . 26
4.3.1 JSF life-cycle scenarios . . . 28
4.4 JSF development process . . . 29
4.4.1 UI Components and the Component Tree . . . 29
4.4.2 `FacesContext` . . . 30
4.4.3 Package `javax.faces.render` . . . 30
4.4.4 Messages . . . 32
4.4.5 Navigation . . . 32
4.4.6 Events and Listener Model . . . 32
4.4.7 Validation Model . . . 34
4.4.8 Converters . . . 36
4.4.9 Backing beans . . . 36

5 JavaPersistence **39**
5.1 Java Persistence API . . . 39
5.1.1 Entities . . . 40
5.1.1.1 Relationship multiplicities . . . 40
5.1.1.2 Entity Life-cycle and Entity Listeners . . . 40
5.1.2 Entity Management . . . 41
5.1.3 Persistence in the Java EE application . . . 42
5.1.4 Persistence in NetBeans . . . 44

6 Java EE Security **45**
6.1 Physical security . . . 45
6.2 Operating system security . . . 46
6.3 Application security . . . 46
6.3.1 Realm/User/Group . . . 47
6.4 Secure Connection Using SSL . . . 47
6.4.1 Java EE application with SSL . . . 48
6.5 Specifying a Security Constraint . . . 49
6.5.1 Specifying a authorized users . . . 49
6.5.2 Resource constraint in Java EE Application . . . 50

II Virtual Center for Educators of Seniors **51**

7 Introduction **53**
7.1 Skeleton plan . . . 53

8 Basic Study **55**
8.1 User Profiling of the Virtual Center . . . 55

8.1.1 Support for multimedia . . . 56
8.1.2 Application Stability . . . 56
8.1.3 Security requirements . . . 56
8.1.4 Internationalization and Localization . . . 56
8.2 Setting specific goals . . . 57
8.3 Use Case . . . 58
8.4 Chosen use cases in higher detail . . . 59
8.5 Usability bounds setting . . . 60

9 Analysis **63**
9.1 Concurrent systems . . . 63
9.2 Data analysis . . . 63
9.3 Technology, frameworks . . . 64
9.3.1 Frameworks . . . 65
9.3.2 Tools . . . 65
9.3.2.1 Netbeans IDE . . . 65
9.3.2.2 CVSNT Server . . . 66
9.3.2.3 StarUML . . . 67
9.3.2.4 DBDesigner . . . 67
9.3.2.5 Freemind . . . 67
9.3.2.6 Inkscape . . . 68
9.3.2.7 Gimp . . . 68
9.4 Server environment . . . 68
9.4.1 Java Sun Application Server . . . 68
9.4.2 MySQL . . . 69

10 Design **71**
10.1 Database design . . . 71
10.2 Component design . . . 71
10.2.1 Page structure . . . 73
10.2.2 Pages description . . . 74
10.3 Graphical user interface . . . 78
10.3.1 Page framing . . . 79
10.3.2 Colors . . . 79
10.3.3 Ordering . . . 80
10.4 Testing . . . 80
10.4.1 Generally about test processing . . . 80
10.4.2 Testing strategy . . . 80
10.4.2.1 Test reports . . . 81
10.5 Acceptance test . . . 81
10.5.1 Conditions of acceptance tests . . . 82
10.5.2 Documentation requirements . . . 82
10.5.3 Functionality requirements . . . 82

11 Implementation **87**
11.1 Self implementation . . . 87
11.1.1 Some Netbeans conventions in Java EE development . . . 87
11.1.2 Login, logout and data security . . . 89
11.1.3 Issues . . . 90
11.2 Unit tests designing and processing . . . 91

12 Testing and installation **93**
12.1 Installation manual 93
12.2 Validation test 95

III Conclusion **97**

13 Conclusion **99**

A Database report **103**

B Screenshots **107**

C Content of CD **113**

Bibliography **115**

List of Figures

1.1 XML annotations in source code . . . 5
1.2 Multi-tiered Application . . . 7
1.3 Model 1 architecture of Java EE applications . . . 8
1.4 Model 2 architecture of Java EE applications . . . 8
1.5 Java EE Platform APIs . . . 10

2.1 CGI script vs. Java Servlet life-cycle . . . 11
2.2 Semantics for inserting listener into DD (*Deployment Descriptor*) . . . 12
2.3 Filter to Servlet Mapping . . . 14
2.4 Filter to Servlet Mapping (Deployment Descriptor) . . . 15
2.5 Filter for all the Servlets . . . 17

3.1 Construction of JSP . . . 20
3.2 Root element in JSP XML compliant document . . . 23

4.1 JSF View tier . . . 26
4.2 Statechart diagram of request processing life-cycle . . . 27
4.3 UI Component and UIComponent Base Classes . . . 30
4.4 Main classes from `javax.faces.render` package in UML . . . 31
4.5 The `javax.faces.event` package . . . 34
4.6 Package `javax.faces.validator` . . . 35
4.7 Fraction from `javax.faces.convert` package . . . 37

5.1 Object-Relational Mapping . . . 39
5.2 Entity Lifecycle . . . 41
5.3 `persistence.xml` example . . . 42
5.4 Persistence entity class example . . . 43
5.5 Simple query on `animal` table . . . 43
5.6 Using JTA transaction manager . . . 44

6.1 Creating user in `file` realm by Admin Console . . . 49
6.2 `security-constraint` element in DD . . . 50

8.1 Primary persona . . . 55
8.2 Negative persona . . . 57
8.3 Virtual Center requirements . . . 58
8.4 General Use Case . . . 59
8.5 Unregistered and Admin user in higher detail . . . 60
8.6 Partition of Use Case for registered user . . . 61

10.1 ER model . . . 72
10.2 General Web application component diagram . . . 73

10.3 Page navigation diagram 74
10.4 Login activities scheme 75
10.5 Diagram shows how is navigated *branch* page 76
10.6 Use Case of *tutorial* page 77
10.7 Graphical user interface partitioning (most of pages) 78

11.1 Code example: *createElearning()* method 89
11.2 Code example: *encodeHeslo()* method 90
11.3 Code example: *prerender()* method 91
11.4 Code example: *logoutButton_action()* method 91
11.5 Code example: *createButton_action()* method 92

B.1 *Login* page 107
B.2 *Hello* page 108
B.3 *Tutorial* page 108
B.4 *Play* page 109
B.5 *User* page 109
B.6 *Elearn* page 110
B.7 *Index* page of vc_jpa application 110
B.8 *List* page of vc_jpa application 111
B.9 *New* page of vc_jpa application 111

C.1 The list of enclosed CDROM 113

List of Tables

1.1 Java EE components 9

3.1 Syntax of JSP directives 20
3.2 Examples of Java Expression Language 22
3.3 Standard Syntax Versus XML Syntax 23

8.1 Table of active user roles 59

10.1 Registration request approving 82
10.2 User suspending 83
10.3 User unsuspending 83
10.4 Switching user roles 83
10.5 Tutorial deleting 84
10.6 Login (admin) 84
10.7 Login (user) 84
10.8 Tutorial suspending 85
10.9 Tutorial unsuspending 85
10.10 Adding new tutorial 85
10.11 Adding e-learning course 86
10.12 Deleting information about e-learning course 86
10.13 Addional tested functionalities 86

11.1 Implemented JSF source components 88

12.1 Summary of validation test results 96

Introduction

Developers today increasingly recognize the need for distributed, transactional, and portable applications that leverage the speed, security, and reliability of server-side technology. In the world of information technology, enterprise applications must be designed, built, and produced for less money, with greater speed, and with fewer resources.

Architectures of today's enterprise applications have become increasingly server-centric. Application functionality which used to be implemented within a fat client application running on client's machine now resides on a central server instance. Responsibilities of the client application are scaled down to presenting the UI (*User Interface*) and to acquite of user's input, making thin client.

By programming of multitiered web applications, developers get into touch with some different technologies, application servers, tools and proceedings. Due to an extent of this topic, some initial knowledges are exacted of readers. Firstly, the reader should be familiarized with the Java programming language. Good way to get to that point is to work through all the basic and some of the specialized trails in [1, 2] and [3]. Next necessary prerequisities are good knowledge of HTML[1] and at least basic imagination about XML[2] technology. The readers, who are not familiar with HTML or haven't at least initial knowledges about XML, are highly encouraged to read some of many books engaged in this topics. By own experience, I recommend these several books about HTML [4, 5, 6] and XML [10, 9]. There is a lot of another quality literature focusing on these areas, therefore I will let the final choice on reader's personal decision.

Objectives

I will make short introduction into Java EE 5 application development process and present basic principles in developing such as for example describing of application life-cycles. As well I will insert some short examples directly into the appropriate sections of the thesis to clarify described theory. Therefore I believe that the thesis will be very interesting and intelligible for Java developers as well as for non Java expert readers like other university students or Java EE beginners. In the second part of the thesis I will design and implement the virtual center for educators of seniors. Fractions of analysis, design, implementation or another part of the project will be presented in this second part of the thesis too. The third part of the thesis will describe achived results.

[1]The HTML (*HyperText Markup Language*) is a predominant markup language for the creation of web pages.

[2]The XML (*Extensible Markup Language*) is a W3C-recommended general-purpose markup language for creating special-purpose markup languages, capable of describing many different kinds of data

Structure of thesis

The thesis itself is divided into two parts. The first part guides a reader through extensive theoretical background of Java EE 5 technology. The second one demonstrates development of specific Virtual Center for Educators of Seniors application. At the end I will summarize whole work. I'll try to evaluate main advantages and drawbacks of Java EE technology in this thesis too. Now I'll make a short overview about the context according to partioning of the thesis into chapters:

- **Part 1 – Theoretical Background** Eight chapters will describe Java EE technology and basic principles used in conjunction with Java EE application development process. Not only about Java EE components but other useful opensource software tools, such as UML modeler, database designer I'll talk too.

  Chapter 1 – Java EE overview describes what is Java EE Application, how looks its architecture like, what is composed from and what about main features are in Java EE version 5 technology.

  Chapter 2 – Java Servlets describes Java Servlet technology, used components, how looks Java Servlet life-cycle, what are listeners and filters and what are the most powerful features of Java Servlets.

  Chapter 3 – JavaServer Pages Technology describes JSP (*JavaServer Pages*), their life-cycle, what are JSP pages composed from or how to use external resources such as JavaBeans components in JavaServer Pages.

  Chapter 4 – JavaServer Faces gives an overview about JSF (*JavaServer Faces*) technology with revolutionary JSF life-cycle. The JSF technology chapter takes the biggest fraction of the theoretical background part of this thesis.

  Chapter 5 – Java Persistence presents Java Persistence technology and especially the part about JPA (*Java Persistence API*).

  Chapter 6 – Java EE Security discusses all the aspects of Java Security and security of web applications in general.

- **Part 2 – Virtual Center for Educators of Seniors** introduces the VC (*Virtual Center for Senior's Educators*) web application as an example for the previously explained technologies and concepts.

  Chapter 7 – Introduction presents the Virtual Center, contain a plan of the project

  Chapter 8 – Basic Study there are the most important requirements gathered and a Virtual Center specification is made.

  Chapter 9 – Analysis contains the analyses of concurrent systems, as well as of possible technological solutions and makes the choice one of them.

  Chapter 10 – Design contains the design part of the project according to the choose technology and application requirements.

  Chapter 11 – Implementation presents the choice of confessed issues as well as some of the most important conventions in Netbeans Java EE development.

  Chapter 12 – Testing and Installation contains short installation manuals and summarize all supported, tested and finished parts of the VC.

- **Part 3 – Conclusion** This part concludes the thesis with a discussion of the concepts proposed and results achieved.

Part I

Theoretical Background – Java EE 5

Chapter 1

Java EE overview

Developers today increasingly recognize the need for distributed, transactional, and portable applications that leverage the speed, security, and reliability of server-side technology. With higher expectations of the customers, higher requirements stay before developers. In the world of information technology, enterprise applications must be designed, built, and produced for less money, with greater speed, and better application performance. To realize such wants, Java EE emerged.

The Java EE defines standard for development enterprise multi-tiered applications and

```
26 @Entity
27 @Table(name = "cd")
28 @NamedQueries( {
29     @NamedQuery(name = "Cd.findByCdId", query =
30         "SELECT c FROM Cd c WHERE c.cdId = :cdId"),
31     @NamedQuery(name = "Cd.findByAuthor", query =
32         "SELECT c FROM Cd c WHERE c.author = :author"),
33     @NamedQuery(name = "Cd.findByTitle", query =
34         "SELECT c FROM Cd c WHERE c.title = :title"),
35     @NamedQuery(name = "Cd.findByYearCd", query =
36         "SELECT c FROM Cd c WHERE c.yearCd = :yearCd"),
37     @NamedQuery(name = "Cd.findByRating", query =
38     "SELECT c FROM Cd c WHERE c.rating = :rating")
39 })
40 public class Cd implements Serializable {
41
42     @Id
43     @Column(name = "cd_id", nullable = false)
44     private Integer cdId;
```

Figure 1.1: XML annotations in source code

provides a powerful set of APIs (*Application Programming Interfaces*), which make development easier and faster than earlier. Simultaneously Java EE 5 uses all benefits from J2SE, such as platform independence, portability across different types of computer's architecture, naming service for access to resources stored in tree type structure or security of application. As well Java EE provides full support for *Java Servlets*[1], *Enterprise JavaBeans*[2] and the other components known in older versions. In Java EE 5 is new support for *JavaServer Faces*[3], server-side user interface component framework for Java technology-based web applications and new API for programming accessibility between Java components and remote data storages. This API, new in Java EE, has been called *Java Persistence*[4].

[1] discussed in chapter 2
[2] discussed in chapter 3
[3] discussed in chapter 4
[4] discussed in chapter 5

Some XML descriptors are now with Java EE 5 optional. Instead, a developer can simply enter the information as an annotation (Figure 1.1) directly into a Java source file, and the Java EE server will configure the component at deployment and run-time.

Java EE enable using the same application on different systems without any modification of source code. Java EE ensures for their components a lot of services, which are provided automatically, such as component life-cycle management or transaction life-cycle management.

1.1 Enterprise Application

The term Enterprise Application is used to describe business-related applications that consist of a server and client part, also referred to as client-server application [11]. As the client part of the enterprise application has to be connected to the application server using a proper communication infrastructure. According to [13], an enterprise application has generally following characteristics:

Multi-Tier Architecture – An enterprise application is separated across multiple tiers, each tier manages one or more specific aspects of the application. More details about architecture of enterprise application will be discussed later in section about application architecture.

Multiple Users and Roles – An enterprise application should support multiple users with multiple rights. To achieve it we define firstly user roles with different privileges. Every user having appropriate user role obtain the same privileges. The behavior and appearance of the application may differ depending on particular role and its privileges.

Confidential Information – An enterprise application manages and works on confidential information not intended to be accessible for the public, e.g. customer's private data, financial information or internal company knowledge. Disclosure of this information rapidly leads to high costs and loss of reputation.

High Availability Requirements – An enterprise application is often business–critical and therefore has to endure heavy loads of requests. Typically, a repetitive short outages of the application results in significant financial and/or reputation loss.

Security Constraint – An enterprise application has typically strong requirements concerning security. This may include the need for authentication, authorization and data integrity. Access to enterprise applications is often not public but restricted to well-defined groups of users like employees, business partners or customers. I will discuss about security questions in the Section 6 a bit later.

1.2 Application Architecture

Java EE platform uses a distributed multi-tiered application model for enterprise applications. Application logic is divided into the separate tires. For better imagination, how tiers cooperate together, look on the Figure 1.2 on page 7.

Logical tiers in Java EE are follows:

- Client Tier – running on the client machine
- Web Tier – running on the Java EE server

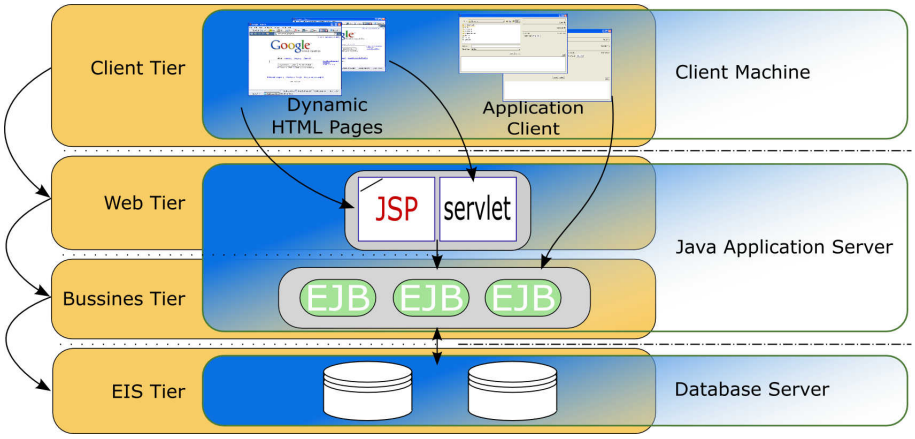


Figure 1.2: Multi-tiered Application

- Business Tier – running on the Java EE server
- EIS Tier (*Enterprise Information System Tier*) – Software component running on the EIS server

Although Java EE applications should consist of four tiers shown in the Figure 1.2, most of the Java EE multi-tiered consist only of three tiers

Very important idea using in Java EE is building applications from components.

1.2.1 Model View Controller

The MVC (*Model-View-Controller*) architecture, which has its background in the Smalltalk environment[5], provides design patterns for developing GUIs (*Graphical User Interfaces*). Application of this architecture leads to a strict separation of the following components:

model represents the business logic part of the application. A clear separation of the model from the presentation logic is essential in order to be able to use different viewing technologies.

view is responsible for rendering model data for client users. Always when data are changed, model notifies view about changes in the model.

controller is the only access point to all actions. When request for another page comes, the controller decides what to show. Based on user actions or results of model updates, the next view is also selected by the controller.

There are several possibilities, how can developer implement these three tiers. Most frequently used are following two ways.

1.2.1.1 Model1

Model1 shown in Figure 1.3 is architecture where presentation and business logic are both implemented by JSP pages (or Java Servlets) and stored data are accessed by JavaBeans classes. Biggest disadvantage of this architecture is junction of presentation and business logic. There is no centralized controller where page navigation could be controlled or the request parameters could be processed, therefore Model1 is a decentralized or page centric approach.

[5]Smalltalk is an object-oriented, dynamically typed, reflective programming language.

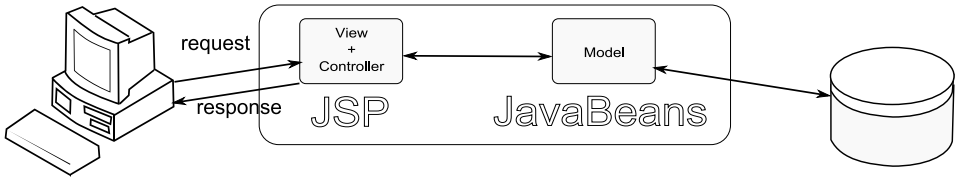


Figure 1.3: Model 1 architecture of Java EE applications

1.2.1.2 Model2

The Model2 design introduces a front controller Servlet as a centralized component that processes the client's requests. It processes the request data sent by the client and selects the next view according to request parameters or state of the model. Components of the view do not refer directly to each other. Since the front controller Servlet is a single point of entry into the web application, it may implement security and logging functions as well. The view can be represented by any presentation technology like JSP or Servlet. The model can be represented by JavaBeans or EJBs (*Enterprise JavaBeans*). This architecture is also called a centralized.

Very good information sources about architecture of Java applications are [14, 24].

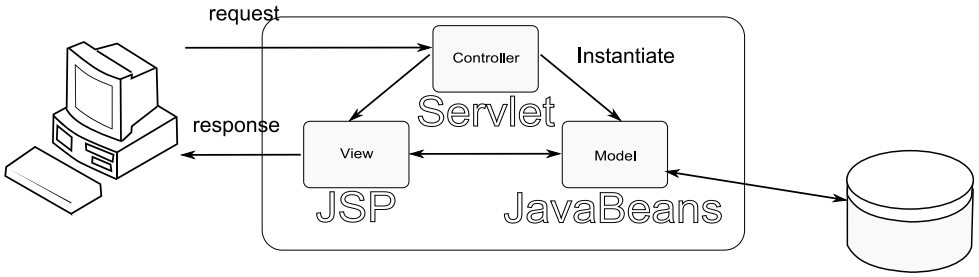


Figure 1.4: Model 2 architecture of Java EE applications

1.3 Java EE Components

Teams of developers create various components, one of which can be for example instrumental to interaction between application and users, another can be instrumental to achieve business logic and so on. Whole application is composed of such Java EE Components. It is possible to reuse functional components during development process and eliminate so production time. Developer can these component naturally put up for sale too. Java EE specification defines components shortly listed in Table 1.1 (page 9).

Note that JavaBeans™ components are not considered Java EE Components by the Java EE specification.

Java EE Clients

Apparently from Figure 1.2 combine with Table 1.1, the Java EE 5 has two essential types of clients.

Application Client An application client runs on a client machine and provides a way for users to handle tasks that require a richer user interface than could be provided by

Component	location
Application component Web component Applet component	Client machine
Java Servlet™ JavaServer Faces™ JavaServer Pages™ (JSP™) technology components	Server machine
Enterprise JavaBeans™ (EJB™) components	Server machine

Table 1.1: Java EE components

a markup language. It means, that application client can communicate directly with business tier running on the application server. Application clients are known as fat clients.

Web Client Web client in contrast with application client has two tasks, parsing HTML and displaying them. All the communication between client and server goes via HTML language. Because of web client's limitations compared to application client, web clients are known as thin clients.

Three server-side components from the Table 1.1 (Java Servlet™, JSP™, JSF™) will be discussed more deeply in separate sections a little bit later.

1.4 Java EE Containers

Containers are interfaces between components and low-level platform-specific functionality that supports the component [14]. Each container hosts only one or more specific application's components and provides standard and container specific Java EE services for contained application components. Following containers are parts of Java EE:

Java EE Server – Runtime portion of Java EE product. A Java EE server provides next two containers (*EJB Container* and *Web Container*)

EJB Container – An EJB Container (*Enterprise JavaBean Container*) hosts EJB components and manages their execution and life-cycle. An EJB Container also provides additional services like transaction control, persistence management and security services to EJBs.

Web Container – A Web container provides network services for executing of JSP pages and Java Servlet components of Java EE applications. Web components and their container runs on "Java EE - enabled" server machine.

Application Client Container – An application client container manages the executing of client components. Application client contains JRE (*Java Runtime Environment*) and runs on the client machine.

Applet Container – Manages running of Java Applets. Applet container consist of Java Applet plug-in integrated in a web browser and the browser. The Applet container runs on client machine.

1.5 Java EE APIs

In Java EE 5 is available fourteen APIs more then in older version J2EE 1.4. It is the biggest difference between J2EE and Java EE. All these APIs can developers use during Java EE application's development processes. Of course there are neither enough place for describing every one of them, nor it is scope of this thesis. On Figure 1.5 is reader able to see, how could Java EE developer user these APIs to achieve user's aims within the enterprise application.

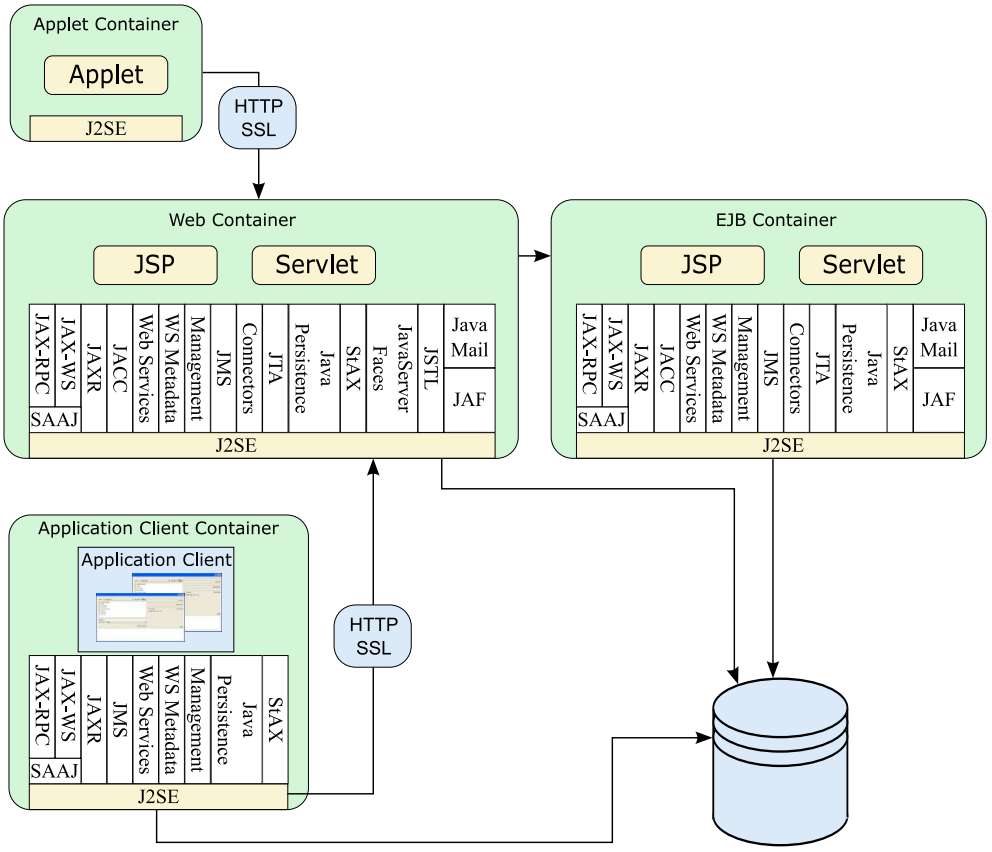


Figure 1.5: Java EE Platform APIs

Following several chapters will discuss about major part of Java EE components and about one new API in Java EE – JPA (Java Persistence).

Chapter 2

Java Servlets

Java Servlets are Java programming language classes that dynamically process requests and construct responses. Every Java Servlet class extends `javax.servlet` interface. Notice that Java Servlet technology is the foundation of all the web Java EE technologies, and so I'll talk very throughly about this topic.

The original specification of first Java Servlet technology (version 1) was created by *Sun Microsystems, Inc.* in the middle of 1997. James Gosling, known as "the father of the Java programming language" first thought about Servlets in the early days of Java, but the concept did not become a product until Sun shipped the Java Web Server product [16]. This was before, what is now the Java Platform Enterprise Edition, was made into a specification.

In the middle of 1990s, dominated the domain of web applications CGI (*Common Gateway Interface*). Biggest drawback of CGI scripts was in bad designed life-cycle. For every request coming over the Internet[1] to server, had to be created new process. This policy can work well only until server exhausts all the available memory. Big problem with defensibility had arisen, when internet community was growing up with staggering rate as in the second half of 90's.

2.1 Servlet vs. CGI

In contrast with CGI, Java Servlets are handled by separate threads within the domain of server process. It means that Java Servlets are also efficient and scalable. Because Java Servlets run within the web server, they can interact very closely with the server to do things that are not possible by CGI scripts [17].

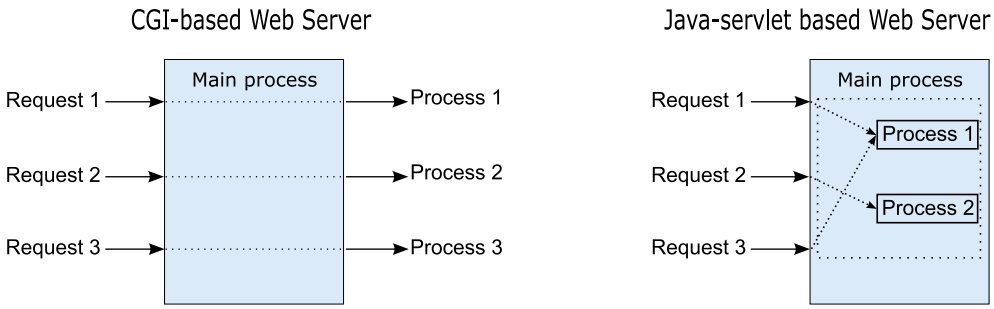


Figure 2.1: CGI script vs. Java Servlet life-cycle

[1] Requests needn't to come over Internet. LAN or another way, such as local access is possible.

2.2 Servlet's life–cycle

In the memory, there is the only instance of the servlet and for every request a new thread is created by container. Developer can change the model by implementing `SingleThreadModel` interface, but this interface is from Java Servlet 2.3 deprecated, and so I'll not take this case into account. Servlet stays in the memory after terminating the request thus other request even other servlets can share such data. Life-cycle of the servlet has three steps:

1. If an instance of the Java Servlet does not exist, the Web Container
   (a) Loads the servlet's class
   (b) Creates an instance of the servlet class
   (c) Initializes the servlet instance by calling the `init()` method
2. Passing request and response objects. (By every users request)
   (a) Creating new servlet's thread
   (b) Invoking the `service()` method – the method firstly detects type of the request (GET, POST...) and then calls method `doGet()` or `doPost()`[2]
3. Finalizing the servlet. (By long-time inactivity or administrator's decision)
   (a) Optional "polite notify" about the intention
   (b) Invokes `destroy()` method

Developers can monitor servlet's life–cycle events by creating special listener objects. Such listeners have to implement `HttpSessionListener` interface. Servlets can register listeners for following life–cycle events:

- Web Context Initialization
- Web Context Destruction
- Web Context Attribute event (added/removed/replaced)
- Session creation/invalidation/activation/passivation/timeout
- Session attribute event (same as previous)
- Request processing started
- Request Attribute event

All the listener classes must be specified in the DD (*Deployment descriptor*)[3].

```
34  <listener>
35    <listener-class>listeners.ContextListener</listener-class>
36  </listener>
```

Figure 2.2: Semantics for inserting listener into DD (*Deployment Descriptor*)

[2]There are another not so frequently used request types – see Section 2.2.2 on page 13)
[3]Deployment Descriptor is XML configuration file for Web technologies in Java Web Applications.

2.2.1 Writing `init()` method

Developer can override the `init()` method of Java Servlet to change its initialization behaviour. Java Servlet, which use Database connection can for example initialize this connection just here. An unsuccessful initialization process throws `UnavailableException`.

2.2.2 Writing `doSomething()` Method

`HttpServlet` interface provides several `doMethods`, that start when the servlet is initialized and any user's request comes. According to Java EE 5 specification, `Method` is one of values: `Delete`, `Get`, `Head`, `Options`, `Post`, `Put` or `Trace` [14]. Which method will be invoked depends on type of incomming request. Selected method will delegate to populate the response.

Indeed, developer can override `service()` method from abstract class `HttpServlet`, but with one restriction. It's necessary to call super.service() method to preserve the functionality of the original `service()` method.

2.2.3 Writing servlet's `destroy()` method

May become, that application server want to reclaim some memory, or simply shut down. For such situations serves `destroy()` method of servlet's objects. This method could be invoked by the application server whenever the requirement to remove a servlet from the memory is arisen, but there is a potentinal risk of data lost when servlets don't finish properly. To eliminate such risk `doMethod()` of servlets has to be implemented as "polite". It means, that no incoming request can be processed until servlet is being in destroying mode, which means that no new threads can be created. Just remain to wait for finishing of all unfinished thread. Threads have to finish properly, or have to be interrupted by admin without any data lost. After that application server can safely remove servlet's object from the memory. If some thread doesn't finish ordinarily, admin user has to interrupt their running "by hand", but such action could cause any data losts or inconsistences.

Usually `destroy()` methods serve to close opened streams or database connections initialized in `init()` method or to start another necessary operation resulting from `init()` or `doMethod()` method of servlet's object.

2.3 Filtering Request and Responses

The Java Servlet specification version 2.3 introduced a new component type, called filters. A filter dynamically intercepts requests and responses to transform or use the information contained in the requests or responses. Filters typically do not themselves create responses, but instead provide universal functions that can be "attached" to any type of Java Servlet or JSP Page [19]. There are the most frequently objectives, which are servlet's filters user for:

- Blocking the request-response pair from passing any further
- Modify the request/response headers and data (object format customization)
- Interact with external resources
- Blocking request from passing (authentication based filter)
- Image conversion (into supported format)
- Data compression (makes downloads smaller)

- Localization (according to user's localization)
- Reformating (to another document format)

Filter is Java EE class, which implements `javax.servlet.Filter()` interface. Because servlet filters are attached from numbers of users, some inconsistences within the application might become. It is also very important to synchronize methods within filter classes. After all the developer mustn't forget to specify filter mapping rules in application's DD.

2.3.1 Creating of filter classes

Most important method in `javax.servlet.Filter` class is `doFilter()`. I'm going to shortly explain, which actions `doFilter()` method performs:

1. Examine the request header
2. Customize the request object
3. Customize the response object
4. Involve the next entity in the filter chain. If current filter is the last in the chain, next entity is the resource at the end of the chain. If current filter is not the last one, next entity is invoked according to configuration tags in DD. Anyway it is very important to call `doFilter` method on the chain object. If no `doFilter` method is called, request will be blocked.
5. When control returns from invoked `doFilter()` method, response headers are examined
6. When any problem during examination has been arisen, exception is thrown

Here is simple diagram showing, how can be filter mapped in the DD including an appropriate XML configuration (Figures 2.3 and 2.4).

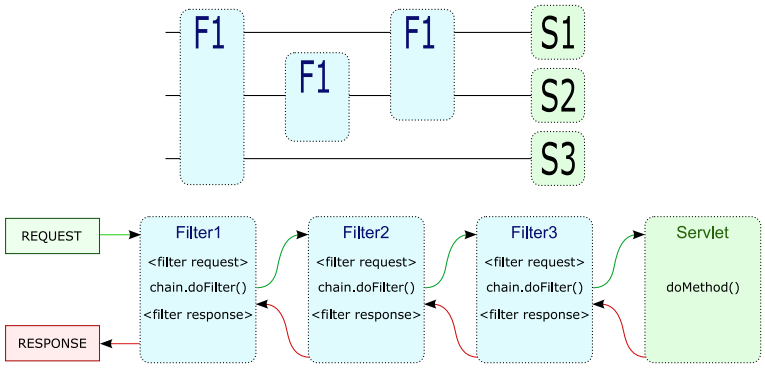


Figure 2.3: Filter to Servlet Mapping

2.4 Constructing of responses

Responses are represented as Java classes implementing the `ServletResponse` interface, that provides many useful operation, which Servlet can use during constructing responses. I'm going to introduce the most frequently used of them.

```
27      <filter>
28          <filter-name>F1</filter-name>
29          <filter-class>filters.F1</filter-class>
30      </filter>
31      <filter>
32          <filter-name>F2</filter-name>
33          <filter-class>filters.F2</filter-class>
34      </filter>
35      <filter>
36          <filter-name>F3</filter-name>
37          <filter-class>filters.F3</filter-class>
38      </filter>
39
40      <filter-mapping>
41          <filter-name>F1</filter-name>
42          <servlet-name>S1</servlet-name>
43      </filter-mapping>
44      <filter-mapping>
45          <filter-name>F3</filter-name>
46          <servlet-name>S1</servlet-name>
47      </filter-mapping>
48      <filter-mapping>
49          <filter-name>F1</filter-name>
50          <servlet-name>S2</servlet-name>
51      </filter-mapping>
52      <filter-mapping>
53          <filter-name>F2</filter-name>
54          <servlet-name>S2</servlet-name>
55      </filter-mapping>
56      <filter-mapping>
57          <filter-name>F3</filter-name>
58          <servlet-name>S2</servlet-name>
59      </filter-mapping>
60      <filter-mapping>
61          <filter-name>F1</filter-name>
62          <servlet-name>S3</servlet-name>
63      </filter-mapping>
64
```

Figure 2.4: Filter to Servlet Mapping (Deployment Descriptor)

Access to output stream via `getWriter()` or `getOutputStream()` method according to type of the stream

Setting content type Operation, which have to be called before response is committed. For more information about content types see IANA[4]

Buffering the output Allows more time for Servlet to set a proper status codes and headers or to forward the control to another component. By default the buffering is disabled. Servlet could buffer its response to avoid sending partial response following by error message.

Setting localization informations (e.g. document encoding)

Status code field for representing status code (e.g. when Servlet cause any exception, or when request is redirected)

Cookies field to store application-specific informations at the client side.

Previous section discussed about filtering requests and responses. There are another two possibilities, how to affect final response.

[4]Assigned Numbers Authority (IANA). Organization dedicated to preserving the central coordinating functions of the global Internet for the public good. [20]

2.4.1 Including other resource

There are two possibilities how to invoke other web resource: directly or indirectly. A web component indirectly invokes another web resource when it embeds URL to another resource. To direct including other resource, an `include()` method of dispatcher object has to be called. To invoke a resource available on the same server you have to firstly obtain a `RequestDispatcher` object by using a `getRequestDispatcher()` method from request object[5].

2.4.2 Transferring Control to Another Component

You might want to partially process a request and then transfer to another component depending on the nature of the request. To transfer the control, invoke the `forward()` method in `RequestDispatcher`[6].

Note that if you have already accessed a `ServletOutputStream` or `PrintWriter` object within the Servlet, you can't use this method, since an `IllegalStateException` would be thrown[21].

2.5 Session management

Every web-based application is responsible for maintaining information based on user's connection. In Java EE, this information are stored in object representing instance of `HttpSession` class and called simply `session`. In Java Servlet Technology exist numbers of principles, that are very important for life-cycle of client's connection. `Session` has usually limited life span set by `setMaxInactiveInterval()` and accessed by `getMaxInactiveInterval()`. As well this value[7] can be set in DD with `session-time-out` element inside of `session` element. An obligation of the server is to associate session objects with connected users. Sessions have usually for this purposes unique identification numbers (session ID). Every user has to know, which IDs are associated to him. Common prodecure is that cookies look after user-side maintaining of this ID. When cookies are at client disabled, ID has to be sent by every request and response. Every hyperlink's url has to be also encoded to contain this ID[8].

2.6 Sharing information

Java Servlet specification apply four types of so called scope objects. Web Context, Session, Request and Page. These objects are used, when the developer want to share some data between more than one component. Shared data are in this case attributes. Each shared attribute (in any scope) has its own `get/setAttribute()` methods like a setter and getter for the attribute. We have to be very rigorous when declaring such sharing object, so that implement synchronized access to all the shared parameters.

2.7 New features in Servlet 2.5

I'm going now explain what new features are in Servlet 2.5 and what are their biggest benefits.

[5] `RequestDispatcher` object can be obtained from either a request or the web context, but usually the request object is used.

[6] See 2.4.1 for information how to obtain `RequestDispatcher` object

[7] Element's value has format of integer, which means how long will session object store until its invalidity in minutes

[8] Servlet has to use `response.encodeURL()` method with every URL

- **Dependency on J2SE 5.0** – new features of J2SE (autoboxing, new enum type, metadata annotation) are guaranteed to developers by programming Java Servlet 2.5.
- **Annotations** – I have discussed about annotations in Chapter 1 a bit. See page 5. Short example is on Figure 1.1.
- **Servlet name wildcarding in DD** – up to date, it is not necessary to insert filter for every servlet separately. Wildcard character is in Java EE allowed (see Figure 2.5 line 18)

```
16 <filter-mapping>
17   <filter-name>Image Filter</filter-name>
18   <servlet-name>*</servlet-name>  <!-- New -->
19 </filter-mapping>
```

Figure 2.5: Filter for all the Servlets

- **Restriction removal** – The specification no longer prevents an error-page handler from producing a non error[9] response.

2.8 Why Servlets overrun CGI

There are the biggest advantages of Java Servlet compared to CGI scripts according to [14, 18] and [17]:

Portability – Servlets are portable because Java is portable (due to byte code and interpreters), and because servlets conform to the well-defined, widely-accepted Java API.

Power – JDK (*Java Development Kit*) is a part of Java EE technology, including all the useful classes and interfaces [10] that developers using by designing standard Java applications. Servlet has access to this classes in the same way like standard Java application.

Efficiency and Endurance – After servlet's initialization by calling `init()` method, resulting object persists over time and over all requests for the Servlet. Actions are effected through method invocation on the object. Multiple concurrent requests are handled by multiple threads on the same object (Figure 2.1).

Scalability – As a result of the previous, Java servlets are highly scalable. Additionally, the fact that there is only one servlet instance regardless of the number of requests means that a servlet can keep track of state information between multiple requests (by using the Session Tracking API).

Integration – Java Servlets are tightly-coupled with the server. A Servlet can use the server to perform duties such as translating file paths. A servlet uses a `ServletContext` object to access server functionality. Some believe that servlet interaction with the server is less error-prone than performing the same actions using server-specific extensions.

[9] Sometimes an error-handling page may be able to do something more graceful than show an error, perhaps choosing instead to show an online help chat window to help the user resolve the problem

[10] InputStreams, OutputStreams, String, Arrays, Threads...

Chapter 3

JavaServer Pages Technology

Servlets make possible to create dynamic content, but by designing the static parts it's necessary to use Java language form. For more extensive content it could be very unpractical and not providing an easy survey. Thanks to JSP (*JavaServer Pages*), developer can write some parts in static form and the dynamic part in Java programming language. This is very effective technique for writing page's web content with dynamic portions.

3.1 JSP's Life-cycle

JSP is before usage converted into Java Servlet form and then services request in the same way as servlets did. Here is the punctual list from JSP Life-cycle behavior:

1. If an instance of the JSP page's Servlet does not exist, the container:
   (a) Loads the JSP page's Servlet class
   (b) Instantiates an instance of the Servlet class
   (c) Initializes the Servlet instance by calling the `jspInit`[1] method
2. The container invokes the `_jspService`[2] method, passing request and response object.

By the first run, the JSP page has to be converted into servlet class, what is very lengthy and resource-intensive process. To eliminate such consumption of server's resources in the next time, it is at first controlled, in case the JSP page is older than appropriate Java Servlet class. If not, JSP page have to be converted anyway. If Servlet is newer than JSP, no conversion is needed and container uses the already created one.

JSP page is composed from standard HTML tags, but including many special JSP tags. In general JSP page can have two different forms. The first form is very similar to HTML with numbers of new tags. The second one is a XML compliant form and in this case, the JSPs are called JSP Documents [14].

3.2 Directives

One of using elements in JSP pages are directives. Directives are used to control how the web container translates and executes the JSP page. Directives are in JSP specification three types: `page`, `include` and `taglib`. By `page` directive written in defined syntax (Table 3.1) is possible to control various JSP page execution parameters. To influence structure of generated

[1] Remember that Servlet classes had `init()` method to do that
[2] According to `service()` method by Servlets

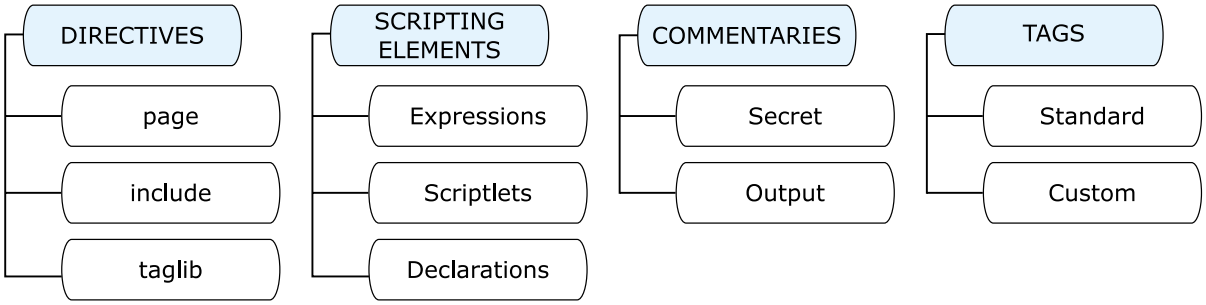


Figure 3.1: Construction of JSP

directive	attribute	possible values
page directive		
<% **page**	language	[Java]
	extends	[package.class]
	import	[package.class | package.*]
	session	[true|false]
	buffer	[none|8kb|sizekb]
	autoFlush	[true|false]
	isThreadSafe	[true|false]
	info	["text"]
	errorPage	["relative URL"]
	contentType	[mimeType [; charset=characterSet] | text/html ; charset=ISO-8859-1]
	isErrorPage	[true|false]
	pageEncoding	[characterSet | ISO-8859-1]% >
taglib directive		
<% **taglib**	uri	["URIForLibrary"]
	prefix	["tagPrefix"]% >
include directive		
<% **include**	file	["relativeURL"]% >

Table 3.1: Syntax of JSP directives

Servlet are used directives `page`. By `page` directive developer can setup, which classes Servlet extends or which libraries should Servlet imports. This directive is often used for setting error page too.

Directive `include` serves inserting other files into JSP page when page is compiled into Servlet. Inserting page can contain JSP construction, which are inserted where directive `include` stayes in the JSP page. (Analogous to preprocessor in ANSI C, which is started before source is compiled).

Directive `taglib` is useful for extending set of available tags. Developer can create own tags, which are modeled from Java classes, that implement special interface. Tags could be associated in TL (*Tag Libraries*), which are configured by descriptors TLDs (*Tag Library descriptors*). So described tags can be used in JSP page after specifying TL by `taglib` directive.

3.3 Scripting elements

Scripting elements provide direct including of source code written in Java programming language into JSP pages. This code is used by compiling into the Servlet, which is invoked by the first run of the JSP page. The first scripting elements I will talk about are expressions.

3.3.1 Expressions

Defined expression syntax is `<%= expression %>`. This expression is evaluated, converted into String type and then into page. Expression is evaluated by every request to JSP page. Values of expression are in generated Servlets stored within `_jspService()`. Other type of scripting element are scriptlets

3.3.2 Scriptlets

It is possible to insert more complex java language code into JSP pages via scriptlets. Syntax of scriptlets is following: `<% Java language code %>`. Scriptlets can pass many of tasks executable with Java. This tasks are done by every request on the JSP page, because source code of scriptlets is included directly into `_jspService()` method. Therefore are all the entities (classes, methods, variables) declared within scriptlets like local.

3.3.3 Declarations

For global declarations exist last type of scripting elements called declaration. Entities declared in global are in generated Servlet outside the `_jspService()` method. In the case of setting that exist the only instance and for requests are created only threads (default), such declarations are available for all requests. Other possibility is to implement any shared entity as static. Syntax for declaration elements follows: `<%! one or many declarations %>`.

3.4 Commentaries

JSP has two types of commentaries. The first one, so called secret commentary, it is a commentary in active JSP code and such comment isn't included into created HTML code. Another type, so called output commentary, is standard HTML commentary and is included into output HTML.

3.5 Tags

Tags are elements in JSP that are associated with some functionality. The biggest benefits of tags are increased readability and separation dynamic and static parts in pages. Still sometimes is composition of static and dynamic elements confusing. JSP technology provides inserting of tags into special libraries, which are used in JSPs by `<@ taglib %>` elements.

Standard tags use prefixies `jsp`. It is possible to use by them special designed classes JavaBeans or include output from another pages into actual page via standard tags. More about standard tags you can find in [15] or [22].

Custom tags use prefixies defined by `taglib` directive and serve for using tag libraries defined by developers.

3.6 Sharing objects

JSP knows four types of scope objects: `request`, `page`, `session` and `application`. These objects are created by container and serves for creating dynamic content. Encapsulation of application's behaviour into objects makes possible for developer to focus on presentation issues. It holds generally, not only for scope objects but for example JavaBeans components. For scope objects, there exist `isThreadSafe` parameter in `page` directive (see in Table 3.1), which can specify, how will container access to such resources. Default setting is `true` and it is the only recommended value, because otherwise this attribute is converted into `SingleThreadModel` which is from Java Servlet 2.3 deprecated.

3.7 JavaBeans Components

JavaBeans components are Java classes which can be easily reused and composed together into applications. JavaBeans components have to contain appropriate `get`*`Property`*`()` methods for every readable property, that returns value of readable property and `set`*`Property`*`()` methods for writing value into appropriate writable property within the JavaBean component similarly with scope objects. In addition JavaBeans classes have to contain constructor that takes no parameters. JavaBean components can be declared in JSP pages with `<jsp:useBean />` component. If referenced bean doesn't exist, the statement creates a new bean and stores its as an attribute of the session scope object. Retrieving of JavaBeans component properties is possible with `${bean.property}` notation or `<jsp:getPropoerty />` element in the JSP page. For setting the property value, there exist `<jsp:setProperty />` element.

3.8 Unified Expression Language

JSF (*JavaServer Faces*) and JSP both have own expression language. The expression language included in JSP provides greater flexibility to the web application developer. JSF is used for rendering UI Components (*User Interface Components*), and need run-time evaluation of

EL Expression	Result
`${1 >(4/2)}`	false
`${4.0 >= 0}`	true
`${100.0 == 100}`	true
`${(10*10) ne 100}`	false
`${'a' < 'b'}`	true
`${'hip' gt 'hit'}`	false
`${4 > 3}`	true
`${1.2E4 + 1.4}`	12001.4
`${10 mod 4}`	2
`${!empty param.Add}`	true if parameter Add is null or a empty String

Unified EL Expression	Result
`#{student.name}`	Gets the value of the property `name` from `student` bean Sets the value of the property `name` from `student` bean during the postback.
`#{student.countCr}`	The return value of the method `countCr()` from user bean

Table 3.2: Examples of Java Expression Language

expressions. Deferred evaluation means that the technology using the unified EL takes over the responsibility of evaluating the expression from the JSP engine and evaluates the expression at the appropriate time during the page life-cycle. JSP EL is designed only for immediate evaluation of expressions.

Another problem is that JSF components need a way to invoke methods on server-side objects during various stages of the life-cycle in order to validate data and handle component events (see Figure 4.2 on page 27). JSP functions are not sufficient because they can be used only for invoking static methods defined in a Tag Library Descriptor (TLD) file; they cannot be used to dynamically invoke public methods on objects. Unified EL was specified for all of these reasons. From Table 3.2 is sure, that standard and unified expression could be for developers useful and working with them is very effective.

3.9 JSP Documents

In short, JSP document is JSP page written in XML syntax. Such documents have all the benefits offered by the XML standard. JSP documents have to be well formed[3]. Most of standard JSP syntax is already XML-compliant, with any exceptions. Elements that are not compliant with their correct XML alternatives are collected in Table 3.3. XML syntax has new element `root`. In general the `root` element is not obligatory in JSP, but may become that it has to be used, because of fulfilment XML compliant form of the whole document. When we have in JSP two root elements (non XML compliant JSP) we can simply change to compliant with wrapping all the text with the `root` like here:

```
<jsp:root version="2.0">
    <!-- our non XML compliant JSP -->
</jsp:root>
```

Figure 3.2: Root element in JSP XML compliant document

Syntax elements	Standard Syntax	XML Syntax
Comments	`<%--...--%>`	`<!--...-->`
Scripting elements	`<%!...%>`	`<jsp:declaration>...</jsp:declaration>`
	`<%=...%>`	`<jsp:expression>...</jsp:expression>`
	`<%...%>`	`<jsp:scriptlet>...</jsp:scriptlet>`
Directives	`<%@ include %>`	`<jsp:directive.include.../>`
	`<%@ page %>`	`<jsp:directive.page.../>`
	`<%@ taglib %>`	`<jsp:directive.taglib.../>`
EL Operator	`<`	`lt`
	`>`	`gt`
	`<=`	`le`
	`>=`	`ge`
	`!=`	`ne`

Table 3.3: Standard Syntax Versus XML Syntax

3.10 JavaServer Pages Standard Tag Library

JSTL (*JavaServer Pages Standard Tag Library*) is set of tags managed by Sun Microsystems, Inc. and develop by the help of Java Community Process. Main objective of JSTL is to provide a tool for for developers with the most frequently used actions by creating JSP pages. Home page of JSTL project is `http://java.sun.com/products/jsp/jstl`. JSTL is divided into four main groups of actions according to their functionality:

1. Core with general functions such as conditions, cycles iteration...

[3]See [9] for more details about XML technology

2. XML for working with XML documents
3. I18N or Internationalization
4. SQL for working with databases

Here is short list about tags defined by JSTL. This list doesn't cover every JSTL tag, but the most commonly used ones. A complete list of the JSTL tags including attributes of them is available in [23].

1. Core (prefix `c`)
   - variable support: `remove`, `set`
   - flow control: `choose-when-otherwise`, `forEach`, `forTokens`, `if`
   - URL management: `import-param`, `redirect-param`, `url-param`
   - Miscellaneous: `catch`, `out`
2. XML (prefix `x`)
   - core: `out`, `parse`, `set`
   - flow control: `choose-when-otherwise`, `forEach`, `if`
   - transformation: `transform-param`
3. I18n (prefix `fmt`)
   - setting Locale: `setLocale`, `requestEncoding`
   - messaging: `bundle`, `message-param`, `setBundle`
   - number and date : `formatNumber`, `formatDate`, `parseDate`, `parseNumber`, `setTimeZone`, `timeZone`
4. Database (prefix `sql`)
   - `setDataSource`
   - SQL: `query-dateParam-param`, `transaction`, `update-dateParam-param`

Chapter 4

JavaServer Faces

After discussing about fundamental technologies and related problematics of Java web applications development, this section focuses on a concrete web application framework JSF (*JavaServer Faces*).

There are today over unbelievable 20 different web application frameworks, and none of these frameworks is based on a standard. So high number was definite indication that a standard solution is missing. JSF have been chosen for this thesis since it is likely that JSF will become the standard solution once it is included into the Java EE platform as proposed by Sun Microsystems, Inc.

4.1 What is JavaServer Faces?

JSF specification says, that it is a server-side UI component (*User Interface Component*) framework for Java technology-based web applications. There are two main components of JSF technology:

JSF API for representing UI components and managing their state

Two custom tag libraries for expressing UI components within a JSP and for wiring components to server-side objects.

JSF UI components are stateful, which means that components keep their state and value across multiple client requests. When using HTML/JSP as the viewing technology for JSF, UI components are mostly a one-to-one mapped to the HTML form elements. This component model also opens up the opportunity for the appearance of RAD (*Rapid application development*) Tools, that allow to develop simply by arranging UI components in a visual development tool. JSF uses Model2 architecture (section 1.2.1.2 on page 8), so there are also benefit of separation of presentation, logic and data. Every from these three parts can be changed without necessity of changing any other part.

4.2 The MVC architecture of JSF

Main drawback of Model2 architecture in compare with SWING Java applications results from the fact that HTTP is a stateless protocol. The only event that is recognized by a web application is the HTTP request, while GUI applications can fire more fine-grained events like a changed value in a text field or a clicked button, mouse moving and so on. A web application has to process all request parameters in order to discover which event occurred. JSF provides a stateful component model in addition to fine-grained events for building event-driven web applications.

Model

As well as it was described in section 1.2 (page 6) about architecture of Java EE applications, model component is composed from business objects, in this case JavaBeans™ components. These JavaBeans may be persisted through an underlying persistence layer and a database through the use of JDO (*Java Data Objects*), Enterprise JavaBeans or an ORM (*Object Relational Mapping*) implementation like Java Persistence (5).

View

Main part of the view tier is a tree of components called with the same name – view. Components can be rendered in different ways depending on client type, but appropriate renderers have to be firstly implemented. Additional delegates are validators and converters, which can be attached

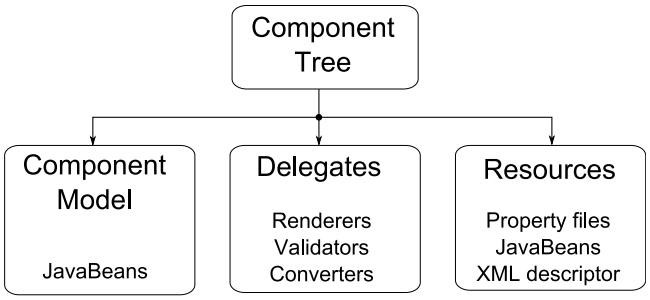


Figure 4.1: JSF View tier

to specific components in order to validate or convert the values entered by the user. Converters provide functionality such as converting data from `String` to another data object type format (e.g. date format, numerical formats ...). Validators check if values delivered by the client are correct, for example if the length of a string is correct. The view also uses resources, that are for example used for localization of the web application.

Controller

The controller in the JSF architecture consists of the front controller Java Servlet class called `FacesServlet`. `FacesServlet` instance accepts incoming requests, passes them to the life cycle for processing, and initializes resources. The configuration file named `faces-config.xml` is a centralized point for managing resources of the web application. It defines navigation according to results of processing in the backend and therefore determines which view is to be rendered next. Notice, that Netbeans IDE uses other conventions and divides `faces-config.xml` into more separate configuration files, `navigation.xml` for navigation rules, `managed-beans.xml` to define scope JavaBean objects.

4.3 The Life-cycle of a JSF Page

JSF has very sophisticated system for control its life-cycle. The whole process consist of six states. If a request is received, an instance of FacesServlet is created. New instance contains whole view structure and represent so current state. Figure 4.2 represents following six steps.

Phase 1 – Restore View

Immediate after the request is received, restore view phase begins. View of the page is made and all the event handlers and validators are wired on appropriate components in the view. The

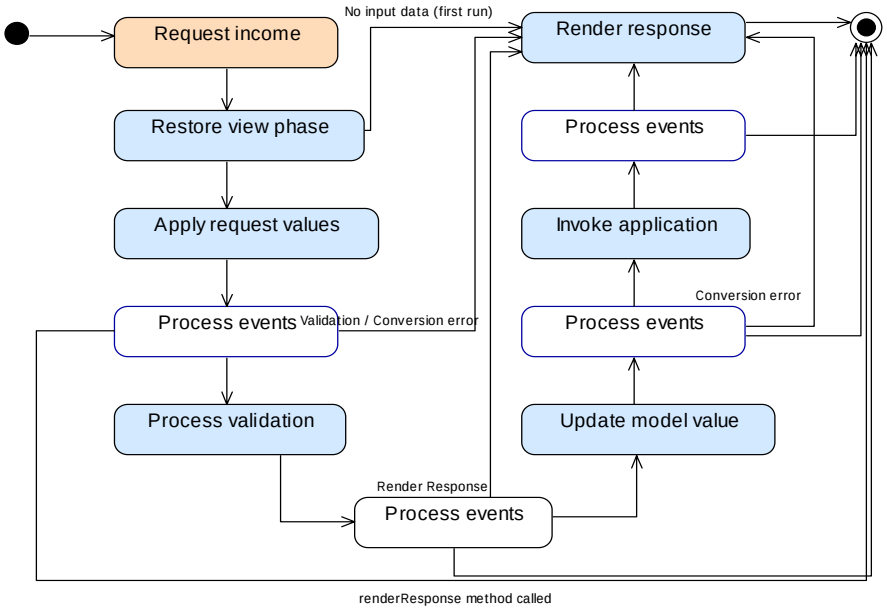


Figure 4.2: Statechart diagram of request processing life-cycle

view is stored in `FacesServlet`. If the request is initial (first request for a page), no request data are available and so another phases are skipped up to render response phase (Phase 6). Empty view is populated. If the request is not initial, view corresponding to this page must exist and it is also accessed.

Phase 2 – Apply Request Values

In apply values phase, every component from current view is accessed and called on decode. If any decode methods or event listeners called `renderResponse()` method on the current `FacesContext` instance, the JSF implementation skips to the render response phase (Phase 6). Input data are converted and stored on components. If data conversion cause error, an error message is generated and queued on `FacesContext`. If event is queued in this phase, JSF broadcast interested listeners. If the application needs to be at the moment redirected to a another web application resource that does not contain any JSF components, it call `FacesContext.responseComplete()`. At the end of the apply request values phase, each input component has an up-to-date value based on the values submitted through the current client request.

Phase 3 – Process Validations Phase

All values are stored in components, but are not validated. In this phase values are compared with specified rules. This may involve component's validators or external validators associated to the components. A negative validation process cause invalidation of related component. After that JSF skips on the render response phase and error message is populated. If any validate method or any listener call `renderResponse()` method on the current `FacesContext`, JSF skips to the render response phase. If the application needs in this phase redirect to be redirected to an other web application resource, `FacesContext.responseComplete()` method should be called. If any listener event arise, interested listeners are broadcasted by JSF.

Phase 4 – Update Model Value Phase

All component values are valid, in proper form and it is also safe to update all values of the backing beans or model objects against the component values. If the local data cannot be converted to the types specified by the bean properties, page is rendered with error displayed. Error message will be the same as while the validation process fails. Any calling `renderResponse()` method will invoke skipping into render response phase. If necessary to redirect to another web resource, `FacesContext.responseComplete()` method can be called. Events are in this phase broadcasted to appropriate listeners.

Phase 5 – Invoking Application

Component values have been converted and validated successfully, all backing beans or model objects have assigned the up-to-date values of the components. Because of this, it is now safe to execute the applications business logic to handle form submissions. Queued events are now broadcasted to interested listeners, for example an event of a button that has been generated during the apply request values phase is now broadcasted. Action listeners contain the business logic themselves or are delegating the execution to other objects.

Phase 6 – Render Response

This is the final phase of the request processing life-cycle. Since all request processing has been completed and the next view is displayed, it is the responsibility of the current phase to render the next view to the client. If this is initial request, the components represented on the page will be added to the component tree as the JSP container executes the page. Otherwise all the component are in the component tree already added, so there is no reason add them again. Components will render themselves as the JSP container traverses the tags in the page. If way to the render response was over any error, error message will be displayed. After the rendering of response current view is stored and so becomes available for next restore view phase.

4.3.1 JSF life-cycle scenarios

In general web applications dont use only JSF pages. Some of pages are static, another are JSP without JSF components. Rendering such web pages depend on type of request incoming from user's. There are logical four scenarios may occur:

Faces Request on JSF content – This is standard JSF scenario, which is described on Figure 4.2

Non Faces Request on JSF content – When user follows any hyperlink in static HTML page, this scenario occurs. Restore view phase creates new component tree and then skips directly to render response phase. No event another user's input is possible, because no active elements in user's request page.

Faces Request on non JSF content – When user requests for an non JSF content (e.g. static HTML page), application will need to redirect to a different web application resource or generate a response that doesn't contain any JSF components by calling `responseComplete()` from `FacesContext` object during apply request value phase of JSF life-cycle.

Non Faces Request on non Faces content – This case is not managed during JSF life-cycle.

4.4 JSF development process

The development of JSF applications is a bit different with classic JSP applications. JSF provides many new features, which are not in JSP and it comes any duties for developers. At first I'm going to say, what are JSF applications composed from, accordingly to that development team must implement particular components. Following short list of components, that must be implemented by developers. Without that reader will be implemented by team of developers:

- JSF pages – Exactly it means JSP pages created by using JSF tag libraries.
- Converters, Validators, Listeners and Backing Beans – we will discuss how to create this custom components a bit later.
- Custom UI components and renderers – will be discussed at the end of this chapter. For common applications are not necessary.
- Configuration of application – It means to configure navigation rules, custom objects and sure creating DD.

4.4.1 UI Components and the Component Tree

UI Components are the basic reusable components for development of UI using the JSF. Unlike Java Swing components, JSF UIComponents are situated on the server-side. This fact leads to a major difference between UIComponents from standard desktop clients and web application user interface components. The latter do not interact directly with the client, so that every interaction between client and server requires a complete HTTP request-response cycle. Compared to for example Java Swing user interface has JSF one big disadvantage. When client inserts any input data for example into text input component, there is no way to validate data at client machine, because all the application logic is stored in program on the server machine. It means, that data must be sent over HTTP to the server, which validate them and answers response. Such communicate calls request-response-cycle.

JSF UIComponents are compoundable, while the composition of components forms a component tree. A component tree or a view is JSF's internal representation of a page. The relations between nested components are a parent-child relationship. For example a form component, as the parent, contains several `textField()` components and labels as children.

The JSF specification defines another kind of relationship between components, called `facet` too. A facet is a subordinate component, that has a named relationship and a specific role according to the superordinate component. The relationship is independent of the usual parent-child relationship. For example a table has a facet relationship to header and footer, while the rows of the table are the child components associated to the table.

The UI component itself defines only its functionality. The appearance to the user is created by a renderer. Therefore, this architecture separates the functionality from the presentation, which allows a flexible handling of different client devices (PC, PDA, Mobile ...).

All user interface components are derived from the base class `UIComponent`, which defines methods for the navigation in the component tree, interaction with the backing data model and managing validation, conversion and rendering.

To simplify the development of custom user interface components a subclass of the `UIComponent` named `UIComponentBase` provides default implementation for all methods, so a developer has only to extend necessary methods without writing the whole component from scratch.

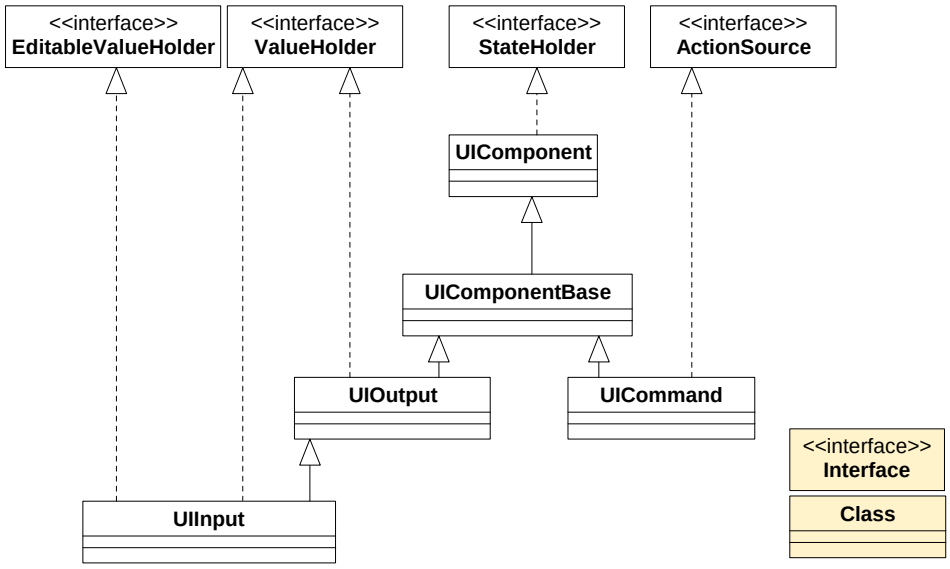


Figure 4.3: UI Component and UIComponent Base Classes

4.4.2 `FacesContext`

The `FacesContext` holds all contextual information about view and defines methods used during the request processing life-cycle. It includes the following elements:

1. `ExternalContext` providing access to the container environment, mainly the surrounding Servlet API
2. `UIViewRoot` represents root of current component tree
3. `RenderKit` is a collection of renderer instances that know how to render JSF UIComponents for specific client
4. The response can be sent either through a `ResponseStream` or `ResponseWriter` object, while the first one is used for returning binary content.
5. Message queue is also accessible by `FacesContext` via `getMessage()` method.
6. `FacesContext` can through method `responseComplete()` terminate request processing as soon as the current phase is completed. This method is used if request was processed in the past and it is also not necessary to repeat it.

4.4.3 Package `javax.faces.render`

Render package includes main classes which makes from tree representation of UIComponents any HTML stream, understandable for web client. There are at least two good reasons, why take interest in renders:

- we want to access JSF by different devices (PDA, PC, text console ...)
- we want to create custom UIComponents, so we have to implement renderers too

What happens when the client device is a mobile phone or PDA that does not provide a HTML browser and therefore requires the web application to respond in another markup language?

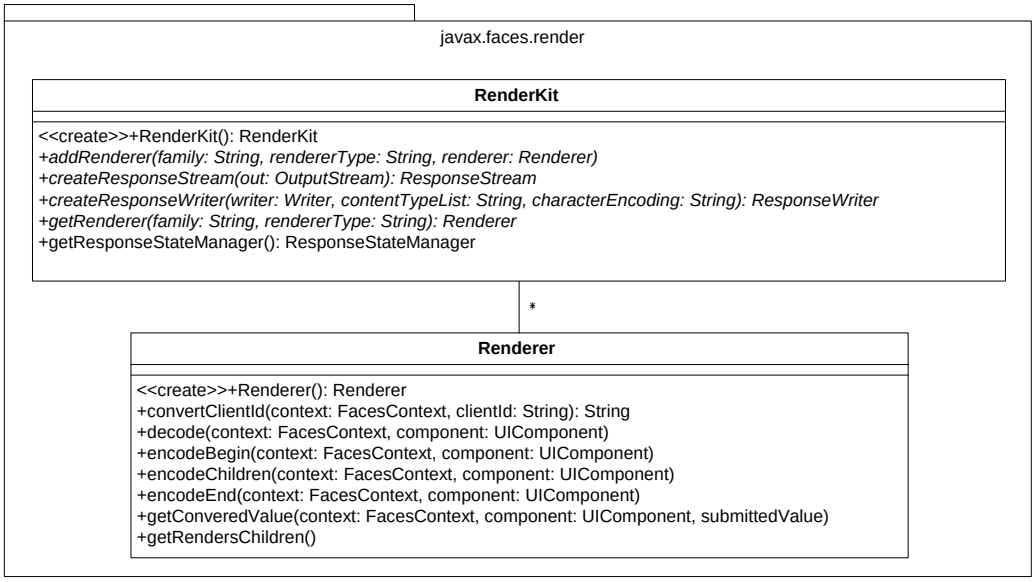


Figure 4.4: Main classes from `javax.faces.render` package in UML

It is also problematic to add support for a different client device later on. It would require a large number of changes within the web application. JSF provides a better solution. Renderer is separate part of JSF application, which has nothing to do with component functionality. A renderer produces the output for one specific UICcomponent which is associated with. A set of renderers is organized into a `RenderKit`. A `RenderKit` is derived from the abstract superclass `javax.faces.render.RenderKit` shown in Figure 4.4.

The rendering model of JSF describes two methods of component rendering:

1. **Direct rendering** encapsulates the rendering logic directly into components, so there is no clear separation of functionality and presentation. To implement direct rendering of a component, the component has to override the `encodeBegin()`, `encodeChildren()`, `encodeEnd()` and `decode()` methods to produce the response to the client itself and handle the data received from the client. These methods are used when a component is only intended to work with a particular client device. It allows to implement a component in one single class, which results in an efficient and compact solution. But this comes at the price of poor maintainability and reusability.

2. **Delegated rendering** uses a separate renderers. It delegates the work of the `encode()` and `decode()` methods to a renderer class. This leads to the full separation of functionality and presentation parts and allows the use of different client types, localization, look & feel and so forth. When using delegating rendering, switching between different clients is just a matter of replacing the `RenderKit`.

A renderer is derived from the abstract superclass `javax.faces.render.Renderer` and implements the same methods a UI component would use for direct rendering. The renderer is a translator between the client- and server-side. If a request is received, the renderer performs the decoding. This means, that the renderer extracts the appropriate values from the request parameters and sets the UI component's values accordingly. In other words, the previous state of the component is reestablished. When JSF is preparing to send a response back to the client, the renderer performs the encoding through creating a representation of the component which the client is able to understand. Most of information about rendering problematics in JSF

pages I gathered from [25].

4.4.4 Messages

Messaging is very important ability, how can be clients informed about state of application. All the time, when client uses application must be informed, what is happening, especially, when any problem occurs. For messaging between server and client JavaServer Faces provides a central mechanism. Messages can be generated from almost every type of component that is involved in the request processing. For example validators, converters, UIComponents and so forth. When a message is generated, it is either associated to a specific UI component like an input field that contains invalid data or it is associated to the whole component tree. JSF maintains a message queue inside the `FacesContext` that stores the messages, so they can be rendered during the render response phase (page 28). Messages are represented through instances of the `FacesMessage` class which contains fields for a summary, a more detailed message and a severity level according to the type of the message.

4.4.5 Navigation

Navigating the client browser between different views based on user gestures is one of the fundamental tasks a web application has to handle. Although it is a very basic task, it can become very complex to keep track of all paths that lead from one view to another. The developer has to spend a lot of work in form of pure Java code or JSTL tags to define which view the application should present to the user, for example after processing the submitted form. Usually there is no central place to configure the flow between the views of a web application. Almost every view has to define the possible paths itself. In a large web application with hundreds of pages, it is almost impossible for the developer to keep up with all possible transitions between the views. Maintaining of such web application and its page flow is a challenge.

JSF provide a navigation system that simplifies the complexity and increases the maintainability of web application's page flow. The main part of this navigation system is the navigation handler, represented through a single instance of the `NavigationHandler` class. The handler is invoked as a result of action events, fired by UIComponents that handle user interaction. These UIComponents are associated with methods on the JavaBeans in the backend. They execute application logic and return an a proper outcome to the result of the processing done by the business logic (dynamic navigation) or return a hardcoded outcome (static navigation). According to this outcome and the navigation rules defined in the JSF configuration file `faces-config.xml`, the next view is selected. Navigation rules define the possible navigation paths. In other words they define which view can be navigated from another view or a set of views according to outcomes of executed methods in the business logic.

4.4.6 Events and Listener Model

Event driven design is very common for SWING applications, but for server-side web applications it is new. The event driven desktop model brings the development to a higher abstraction level, removing the need to operate directly on level of the underlying stateless request-response model of HTTP. Therefore, integration of the application logic is not more than assigning listeners to UIComponents that generate events appropriate to the listeners.

Events represent user interactions with the UI and therefore provide the main mechanism for the UIComponents to propagate user actions. Events are for example triggered by clicking a button or changing a value in some input field of the UI.

The eventmodel of JSF is based on the eventmodel, which is also the basis for the Java Swing GUI toolkit. JSF technology supports three kinds of events:

- Value-changed events
- Action events
- Data-model events

A value-change event occurs when the user changes the value of a component represented by `UIInput` or one of its subclasses. For example it should be done by selecting of any checkbox, also an action which results in the component's value changing to true. UIComponents, which can generate these types of events are `UIInput`, `UISelectOne`, `UISelectMany` and `UISelectBoolean` components. Value-changed events are fired only when no validation errors were detected. Depending on the value of the immediate property can be these events processed in different phases of the JSF's life-cycle. Action events can be processed during the invoke application phase or the apply request values phase and value-change events can be processed during the process validations phase or the apply request values phase.

Action event occurs when the user activates a component which implements the `ActionSource` interface. These components include buttons and hyperlinks.

Data-model events occurs when data components process a row of their data. Unlike the previously named events, datamodel events are not fired by user interface components. Classes derived from the `javax.faces.model.Datamodel` class allow the registration of listeners those take care of handling data-model events. Since it is not a UI component the listeners are associated to, the registration can only be done in the Java code, rather than binding it to a component in the view. The `Datamodel` class is a wrapper around different data binding technologies, with its underlying data modeled as a collection of objects that represent the data rows. This wrapper allows to use common data source objects, such as a JDBC ResultSet or a JSTL Result.

The development of JSF listeners consist of:

- Implement own event listener class
- Implement method of backing bean which will handle events and refer to this method via an expression associated with attribute of the component's tag.

All events are queued in the `FacesContext`. After each phase of the request processing life-cycle, events with the appropriate PhaseId of the current life-cycle phase are broadcasting to registered listeners. It means that the appropriate event handling methods are invoked on the listeners.

It is important to distinguish between the event model of a standard desktop and a web based application. The latter always needs a request-response cycle for recognition events. Therefore it is not possible to react in the server-side immediately to user's input.

Figure 4.5 shows the classes and interfaces of the `javax.faces.event` package. It includes the standard events and listeners defined through the JSF specification. `FacesEvent`, `FacesListener` and `EventListener` do not contain any methods that have to be implemented and are therefore called tagging-interfaces. These are only used for grouping of subclasses in an object hierarchy.

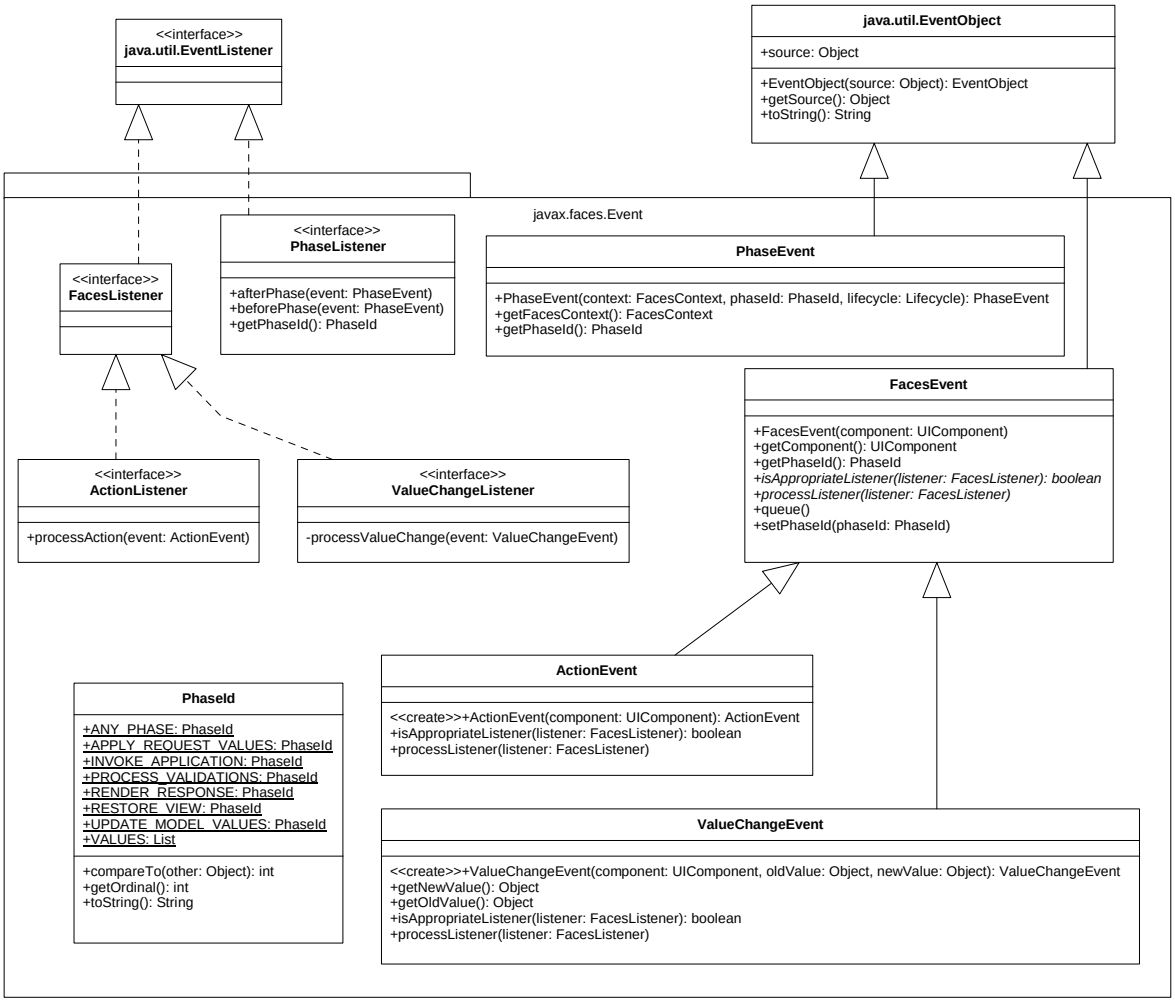


Figure 4.5: The `javax.faces.event` package

4.4.7 Validation Model

Validation Model is JSF support mechanism for validating the local data of editable components such as text fields. Validators are used to prevent application in improper behavior. Additionally many of the security and stability problems in web applications are caused by incorrect format of input data submitted from the client. On Figure 4.6 you can see all standard validation classes.

One or more validators may be associated to a UI component that implements the `EditableValueHolder` interface defined in the JSF API. This means that the component can hold a value, like e.g. `textFields`. A validator associated with a component checks the local value of the component during the process validations phase (page 27 Request processing lifecycle).
Validation can be implemented by developer in two ways:

1. Direct validation can be simply implemented by overriding `validate()` method of the UIComponent superclass. Validation logic is encapsulated into UIComponent, and therefore is not usable by other UIComponents.

2. Delegated validation provides two types for implementation of validators. One possibility

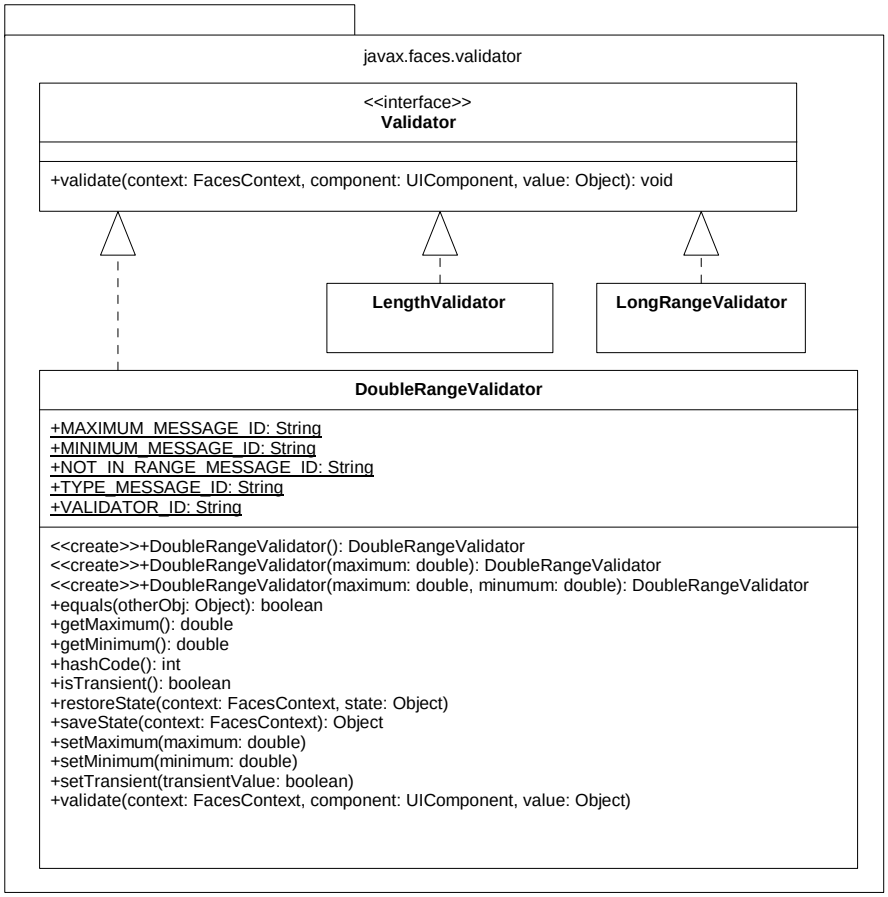


Figure 4.6: Package `javax.faces.validator`

is to implement `Validator` interface that performs the validation process. Developer has to implement constructor method, a set of accessors methods for any attributes on the tag and a validate method, which overrides the validate method of the `Validator` interface. After implementing `Validator` interface it remains to register the validator implementation in the application and to create custom tag or use validator tag to register validator on the component. Other possibility is to delegate the validation to a method of a JavaBeans through a method binding expression.

Validation is processed during validation phase of the JSF life-cycle. When any validator checking fails, an exception is thrown and UIComponent get the information about of their value's invalidity. Rest of the life-cycle phases are skipped because there is a potential risk for the data integrity when the data submitted by user are invalid. An error message notifying user about wrong input is appended to the response.

Another possibility to validate user's data is to implement JavaScript[1] code, which will validate data on the client's side. The only advantage of this method is decreasing of necessary request-response cycles, because data needn't to be sent on server for validation. The major drawback of client-side validation is, that a lot of web clients don't support Java Scripts. It is not sure, that the data would be validated.

[1] JavaScript is a prototype-based scripting language with a syntax loosely based on C. The language is best known for its use in websites, but is also used to enable scripting access to objects embedded in other applications.

4.4.8 Converters

A converter is used to format an object to a "nicer" text to be displayed. For example if you want to display a date in JSP you can use a converter to reformat the date in form, which user understands, like "10/03/2005".
But there is another way to use a converter. If you use them in conjunction with an input control, the user's input must be in the format specified by the converter. If the format of the input doesn't match the format you can throw an exception in the converter which could be displayed to the user. The associated object will not be updated. There are also two different viewpoints that is handled by web application. Model view, which represent data in business logic and presentation view, which represent the same data for user by `String` format.

JSF technology provides numbers of converter, which can be used for conversion data from presentation form into model and vice versa. Converter components implement the `javax.faces.convert.Converter` interface defined by the JSF specification. It consists of two methods, each one in charge of converting in a particular direction:

- `getAsObject()` – returns `Object` representation of current data
- `getAsString()` – returns `String` representation of current data, using for presentation to the user

Since converters are pluggable components which can be associated to many different components, they provide an optimal reusability. JSF defines a set of converters, for example for date, time and numerical conversions. Figure 4.7 shows converter classes defined inside of `javax.faces.convert` package.

4.4.9 Backing beans

Typical JSF application includes one or more backing beans. Backing beans are JSF managed beans that are associated with UI components used in a particular page. Backing beans are Java classes that contain properties and methods for storing and manipulating the user's data. The event handler methods can manipulate the user interface or execute application logic in the backend. Each of the component properties can be bound to one of the following:

- A component's value
- A component's instance
- A converter instance
- A listener instance
- A validator instance.

The most common functions that backing bean method perform include the following:

- Validating of component's data
- Handling of an event fired by a component
- Processin to determine the next page to which the application must navigate – Navigating

When a bean property is bound to a component's value, it can be any of the basic primitive and numeric types or any Java object type for which the application has access to an appropriate converter. Also property can be `Date` type if the application has access to a converter that can

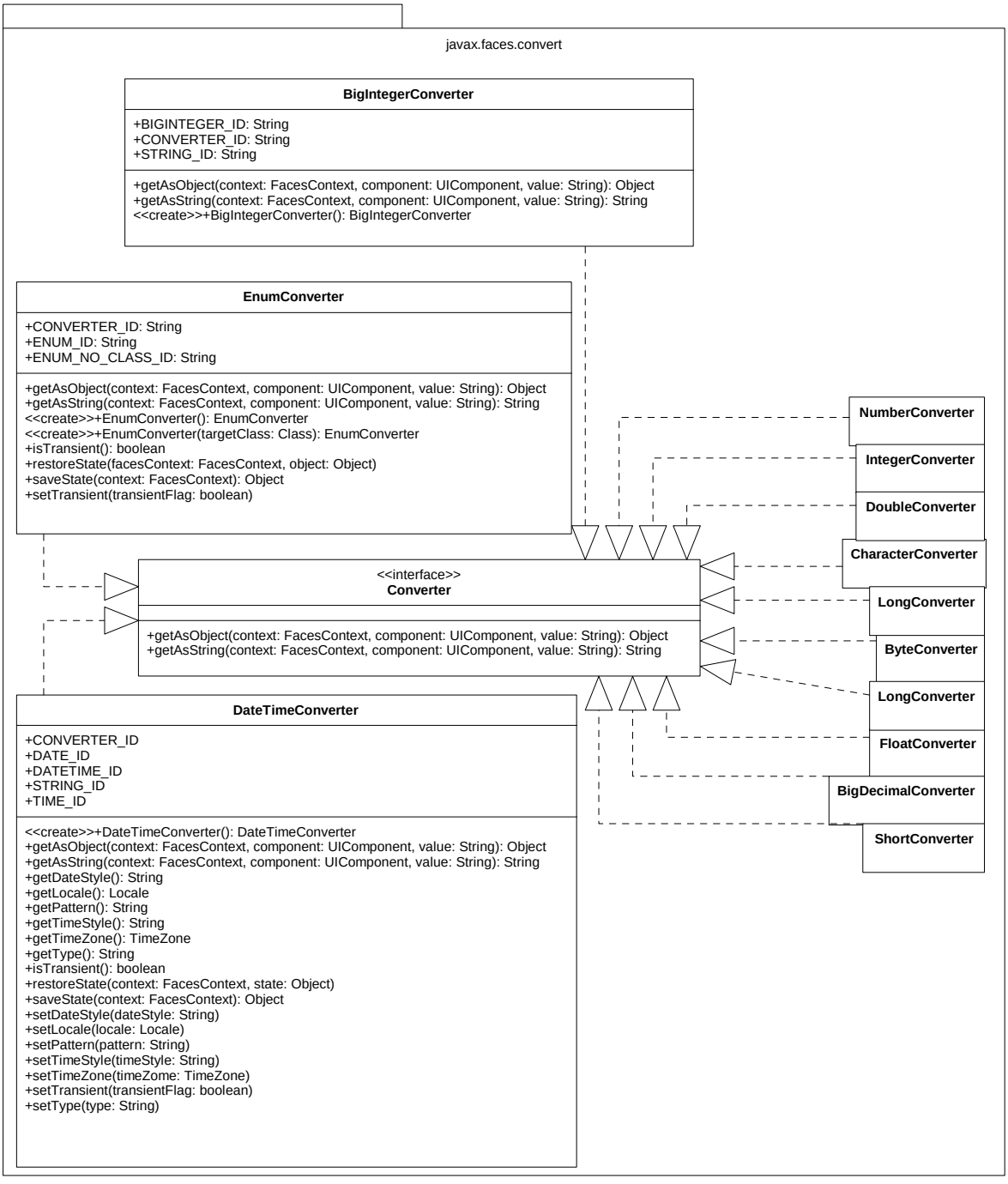


Figure 4.7: Fraction from `javax.faces.convert` package

convert the `Date` type to a `String` and back again. If the property is bound to a converter, validator or listener instance then the property has to be of the appropriate converter validator or listener type.

Managed Beans, which are a declarative method of creating backing beans and model objects release the developer from the task of creating backing beans manually. A backing bean that is declared in the configuration file as a managed bean is made to be available to the application throughout the application life-cycle. The backing beans are stored in the scope, according to the specification in the configuration file.

Chapter 5

JavaPersistence

Java Persistence API is a new API providing object-relational mapping for accessing relational databases by Java EE 5 applications. Last chapter discussed about JSF – one possibility, how to create dynamic web content with higher level of abstraction. Although web development advances from pure scripting for the generation of dynamic content to a more object oriented approach, the underlying data sources are still at most relational databases. As a result, moving the presentation layer to an object oriented approach results in a mismatch between the relational model of the databases and the object oriented view in the presentation layer and possibly also in the business layer. Java Persistence is not only an API for mapping such mismatched data models, but it is complex Java EE conception containing Java Persistence API and the query language.

5.1 Java Persistence API

Nowadays, the most accepted solution to bridge the object-relational gap is a technique called ORM (*Object-Relational Mapping*) that mediates between the object oriented view of the application and the relational tabular data representation of the relational model. Tabular data is mapped to persistent Java objects, which allow data access in form of objects containing the data of a particular row in the database. Java object that maps to a database table is called an entity class. It is a regular JavaBeans component (also known POJO, *Plain Old Java Object*) with properties that are mapped into columns in the database table. To manage the interaction

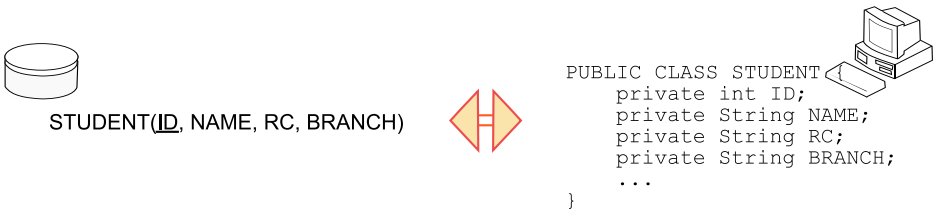


Figure 5.1: Object-Relational Mapping

of entities with the Java Persistence facility, applications use the `EntityManager` interface. The set of entities, that can be managed by an entity manager are defined in a persistence unit, which is configured in `persistence.xml` file. In this section I'll describe how create link such as you can see on the Figure 5.1. Following sections will discuss terms connected with ORM.

5.1.1 Entities

An entity typically represents a table in a relational database and each entity instance corresponds to a row in that table. Persistent state of an entity is represented either through persistent fields or persistent properties. All the field and properties within entity class use object relational mapping annotations to map the entities and entity relationships to the relational data in the underlying data store. Each entity must have one unique primary identifier. Field, which represent primary identifier is annotated by `javax.persistence.Id`. Primary key can be according to Java Persistence specification simple or composed from more then one properties of entity class.

5.1.1.1 Relationship multiplicities

There are four multiplicities supported by Java Persistence:

One to one : Each entity instance is related to single instance of another entity. One to one relationship uses `javax.persistence.OneToOne` annotation on the corresponding persistent field. For example one employer can be related with one office.

One to many : Entity instance can be related to more then one instances of another entity. This multiplicity uses `javax.persistence.OneToMany` annotation. An example could be customer, who has more than single contact record.

Many to one : When multiple entity instances can be related with one instance of another entity, property field is annotated with `javax.persistence.ManyToOne`. An example can be more students related to single class of school.

Many to many : The entity instances can be related to multiple instances of each other. For example in college each course has many students and every student may take several courses. Property field has to be annotated by `javax.persistence.ManyToMany`.

Each from relations multiplicity can be in two directional mode. According to owning side of a relationship we recognize these types:

1. Bidirectional relationship: each entity has a related field or property that refers to the other entity. For example if a faculty knows what department instances it has and if departments know what faculty they belongs to, then they have a bidirectional relationship.

2. Unidirectional relationship: only one entity has a relationship field refers to another entity. For example an order knows, what product it orders, but product doesn't know, which order is ordered by.

Sometimes some entities are dependent on existence of some another entity. For example some project entity has a files entities, but when a project is removed, files should be also removed. Such type of dependence is known as cascade and we have an opportunity such dependence in Java Persistence denote by `cascade` element.

5.1.1.2 Entity Life-cycle and Entity Listeners

Such as JSF or Java Servlets, entities have their own life-cycle. Each entity can hold one of four states:

- New Entity: Entity created by using `new` keyword. New Entity doesn't have persistent representation.

- Managed Entity: Entity having persistent identity and associated with persistence context.

- Detached Entity: Detached Entity has persistent identity, but it is no longer associated with an persistence context.
- Removed Entity: Has persistence identity at the moment, is associated with persistence context, but is scheduled for removing from database.

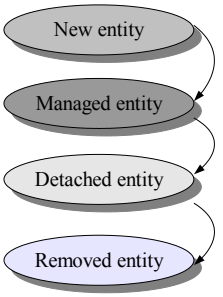


Figure 5.2: Entity Lifecycle

Entity Listeners

Entity listeners or callback method are designated to receive invocations from persistence provider at various stages of entity lifecycle. Callback methods annotate callback handling methods right in the entity class or put them in a separate listener class. Each event has designed appropriate annotation:

- PrePersist or PostPersist : `@PrePersist` or `@PostPersist` annotation marks an entity's method as a callback before or after a new entity is created in the database.
- PreRemove or PostRemove : `@PreRemove` or `@PostRemove` annotation marks an entity's method as a callback before or after an entity is deleted from the database.
- PreUpdate or PostUpdate : `@PreUpdate` or `@PostUpdate` annotation marks an entity's method as a callback before or after updates to an entity are saved to the database.
- PostLoad : `@PostLoad` annotation marks an entity's method as a callback after the entity is loaded from the database.

5.1.2 Entity Management

To manage the interaction of entities with the Java Persistence facility, applications use the `EntityManager` interface. This interface provides methods that perform common database functions as querying and updating the database. The set of entities that can be managed by an entity manager are defined in a persistence unit. The persisting unit is defined and configured by `persistence.xml`. An application that uses the JPA does not need to explicitly create a connection to the data source, as it would when using JDBC[1] technology exclusively. The only condition which has to be performed for using database connection by JPA is created DataSource object in the Application Server[2]. In the next section I'll try to demonstrate, how to work with Java Persistence without any advanced tools to make better sense of Java Persistence setting and working with.

[1]JDBC (*Java Database Connectivity*) API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases – SQL databases and other tabular data sources, such as spreadsheets or flat files.

[2]When using NetBeans IDE combined with Sun Java Application Server, DataSource is configured automatically by deploying project phase.

5.1.3 Persistence in the Java EE application

For simplicity I'll presume, that I have installed MySQL server where "MyDB" database has been already created. "MyDB" will contain for example table of animals in ZOO Troja in Prague and another table with pavilions ibidem.

Creating a DataSource

Creating of DataSources phase depends on using Application Server. In addition DataSources and work with Application servers in general is not aim of this work, and so, I'll refer readers to some resource engaged with using application server. In the following text I'll presume created data source with name jdbc/BookDB. A simple tutorial about adding connection pool and JDBC data source to Application Server is a component of installation manual in second part of thesis (93).

Defining Persistence Unit

Persistence Unit is defined by `persistence.xml` file which is packaged with the application WAR file. This file includes following XML elements:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="1.0"
3              xmlns="http://java.sun.com/xml/ns/persistence"
4              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5              >
6   <persistence-unit name="MyDBPU" transaction-type="JTA">
7     <jta-data-source>jdbc/myDB</jta-data-source>
8     <properties/>
9   </persistence-unit>
10 </persistence>
```

Figure 5.3: `persistence.xml` example

- `persistence` element that identifies the schema which descriptor validates against and includes a persistence-unit element
- `persistence-unit` element defines the name of a persistence unit and transaction type
- `description` element is optional and contains description of persistence unit
- `jta-data-source` element specifies global JNDI name of the JTA data source. Transactions will be managed by container.

Creating entity classes

Entity class is JavaBean component including special Java Persistence annotations and represent a table in the database. Entity class must implement `Serializable` interface. Referenced annotation elements are:

- `@Entity` is the annotation for entity class
- `@Id` is the annotation for selection which property represents the primary key of the table
- `@Table` optional annotation to identify the name of the database table if it is different than the name of the Entity class

```
20 @Entity
21 @Table(name = "animal")
22 @NamedQueries( {@NamedQuery(name = "Animal.findByAnimalId")})
23 public class Animal implements Serializable {
24
25     @Id
26     @Column(name = "aId", nullable = false)
27     private Integer aId;
28
29     @Column(name = "aName", nullable = false)
30     private String aName;
31
32     @Column(name = "aDescription")
33     private String aDescription;
34     ...
35 }
```

Figure 5.4: Persistence entity class example

Entity class "Animal" (on Figure 5.4) contains three private attributes: `aId`, `aName` and `aDescription`. All these variables has in class own setter and getter methods. In addition there are necessary imports of `javax.persistence.*` and `java.io.Serializable` packages because of `Serializable` interface and persistence annotations are used within.

Obtaining access to an Entity Manager

I said, that for managing interaction between entity classes and database application is used Entity Manager. But there is a question, how to obtain access to Entity Manager? Entity Manager is managed by container in Java EE server. There must also exist any instrument, how to obtain Entity Manager for remote clients as well as for JavaBeans components. It is logical, that remote client can not create new instances of Entity Manager, because Entity Manager is for him a remote object. So in Java Persistence exist object Entity Manager Factory, which can be injected by using `@EntityManagerFactory` annotation in requesting object. From Entity Manager Factory can be yet obtained Entity Manager instance.

This technique can be used only for container managed components such as JavaServer Faces for example. For non-container managed objects developer has to include `Servlet` and `ServletContextListener`. These object can then give the application's beans access to the resources by using `@PersistenceUnit` annotation.

Accessing Data from the Database

We have already obtained EntityManager, so we can work directly with database. There will be described, how to create simple database query on database.

```
50     public List getAnimals() throws BooksNotFoundException {
51         try {
52             return em.createQuery("SELECT an FROM Animal an ORDER BY an.aId")
53                     .getResultList();
54         } catch (Exception ex) {
55             throw new BooksNotFoundException(
56                     "Could not get books: " + ex.getMessage());
57         }
58     }
```

Figure 5.5: Simple query on `animal` table

Simple query on Figure 5.5 will extract from `animal` table all records ordered by value of ID. Resulted list will be returned as `List` of records.

For changing value in database we have to use a appropriate setter (because "aName" parameter is defined with `private` access right) method in "Animal" class and change parameter. For example with `an.setAName(String newName)` can be changed a Name of the animal.

`UserTransaction` resource

To ensure that the update is processed in its entirety, the call to `changeName()` method that calls `setAName()` method can be wrapped in a single transaction. Following code is a example of using `UserTransaction` interface by `@Resource` annotation.

```
@Resource
UserTransaction utx;
...

try {
    utx.begin();
    animalDBAO.changeName(animalName, newName);
    utx.commit();
} catch (Exception e) {
    try {
        utx.rollback();
    } catch (Exception e) {
        System.err.println("Rollback failed:");
    }
}
```

Figure 5.6: Using JTA transaction manager

5.1.4 Persistence in NetBeans

NetBeans from 5.5 has special templates for creating Entity classes directly from database. Developer chooses only data source and selects for which database tables want to create entities. Then must insert persistence unit, which will be used, or can generated the new one. All entities will be created. Very good tutorial how to use NetBeans for database connection in Java EE applications can be find in [30].

The Virtual Center application discussed in second part of this thesis is not implemented using JPA. Visual Web Pack module, a new Netbeans module for Java EE web application development is determined to use `CachedRowSetXImpl` classes for accessing database.

The thesis contain enclosed application vc_JPA that doesn't use UI component and wasn't developed in VWP. `vc_JPA` application you can find in `/vc_JPA/` enclosed CDROM.

Chapter 6

Java EE Security

Security of application is today one of the most important questions associated with web application development. Possible outflow of sensitive data from developed application to the prejudice of third party could cause big monetary costs but above all irreversible lost of reputation too. This chapter will discuss, how to make application as secure as possible with Java EE 5 technology.

This chapter will not discuss about the failure of human element, but purely about Java EE security technology, question of security of any Java EE application. Some areas of security depend on used application server and can be a bit different by using another application server then Sun Java Application Server PE 9. At first I list a few basic terms associated with security:

Authentication – The means by which communicating entities prove to one another that they are acting on behalf of specific identities that are authorized for access. Java EE differentiate client-, server- and mutual-authentication.

Authorization – Authorization is based on identification and authentication and provides controlled access to protected resources. Authorization ensures, that users have permission to perform operations or access data.

Data integrity – Recipient of data sent over an open network must be able to detect and discard messages that were modified after they were sent.

Confidentiality – Confidentiality is used to ensure that informations are made available only for users who are authorized to access it.

Non-repudiation – Mechanism used to prove that a user, who performed some action, can not reasonably deny having done so.

QoS (*Quality of Service*) – The system and system data have to be available when needed.

Auditing – System maintaining a record of transactions and security information.

6.1 Physical security

The computer running the web server should be kept physically secured in a locked area. Any backup storage media (tapes, removable disks, etc.) should be similarly protected.

6.2 Operating system security

Application Server runs on any physical server on operation system. To eliminate risks associated with operation system security, developer and administrator of the server must take into account following fundamentals:

- **Limited services** – The services offered by the computer running the web server should be kept to a minimum. This minimizes the threats to the web server, since each network service carries its own risks. By eliminating all nonessential services you eliminate potential holes through which an attacker could break into your system. Examples of services which may pose un-needed risks include mail, FTP, file sharing, remote access, etc.

- **Most privileged user** – The number of users with most privileged access (e.g. root in UNIX or Administrator in NT) should be kept to a minimum. The most privileged users must never use cleartext, re-usable passwords for remote authentication since passwords can easily be sniffed over public networks.

- **Limited number of account** – The number of user accounts on the system should be kept to a minimum. This minimizes threats because it limits the number of accounts capable of attempting to elevate privileges without authorization.

- **Authentication** – If weak authentication (i.e. re-usable, cleartext passwords) is to be used for unprivileged accounts, then user passwords must be at least seven characters long; must not be dictionary words; must contain a mix of alphabetic, numeric and special characters and must change at least every sixty days. Good password security is the first line of defense against system abuse. Intruders will often try to guess passwords or will try to crack them after stealing the encrypted password database.

- **Platform-specific risks** – Most operating systems are insecure by default when they arrive new, out-of-the-box. Vendors of operating systems and application software regularly patch issues to fix serious security weaknesses in their software. Security patches must be applied on a timely and ongoing basis.

- **Logs** – Logs help ensure accountability. Knowledge that logs are kept acts as a deterrent to abuse. Logs are also essential in investigating incidents after the fact. Logs are typically created both by the operating system as well as by applications like web servers, mail servers, etc. To ensure integrity, logs should be written to another computer whenever possible. Logs often contain sensitive informations such as dates and times of user accesses. Logs containing sensitive informations should be accessible only by authorized staff and should not be accessible as public.

6.3 Application security

Securing containers

Java EE containers are parts of Application Server and are responsible for providing application security. In Java EE, containers ensure two types of security: declarative and programmatic. Declarative security expresses an application component's security requirements using DD. In our case DD for web application is named `web.xml`. Programmatic security on the other side is embedded in an application and is used to make security decisions. Programmatic security is useful above all when declarative security alone is not sufficient to express the security model of an application.

6.3.1 Realm/User/Group

Authorization process is based on identification and authentication. Identification is a process which enables recognition of an entity by a system, and authentication is a process that verifies the identity of a user, device, or other entity in a computer system, usually as a prerequisite to allowing access to resources in a system.

Realm

For a web applications, a realms are a complete database of users and groups that identify valid users of a Web application (or a set of Web applications) and are controlled by the same authentication policy. Java EE recognize three types of realms [32]:

`file` – server stores user credentials locally in a file named `keyfile`. For managing users in `file` realm can be used Admin Console. This realm is used for the authentication of all clients except for web browser clients that use the HTTPS protocol and certificates.

`certificate` – server stores user credentials in database. When using `certificate` realm, the server uses certificates with the HTTPS protocol to authenticate web clients. To verify the identity of a user in the `certificate` realm, the authentication service verifies an X.509 certificate.

`admin-realm` – server stores user credentials locally in a file named `admin-keyfile`. For managing users in `file` realm can be used Admin Console in the same way as with `file` realm.

User

Java EE application users are Similar to operating system users an individual identify that has been defined in the Application Server. Users can be associated with a groups.

Group

For example in enterprise information system can most of users belong to the employer group, managers in manager group, directors to director groups and administrator to administrator group. Groups make easier to control the access of large numbers of users.

Role

A role is an abstract name for the permission to access a particular set of resources in a frame of some application. A role can be compared with a key that can open a lock. There is not essential, who you are, but what about the role you are associated with. Security roles for applications are defined are defined in the Java EE DD file `web.xml`, and the corresponding role mappings in the Application Server deployment descriptor file `sun-web.xml`.

It is irrelevant for developers, which categories in realm will be defined for the application. Java EE platform, the security architecture provides a mechanism for mapping the roles defined in the application to the users or groups defined in the runtime realm. For such role-mapping is used `security-role-mapping` element within runtime DD (`sun-web.xml`).

6.4 Secure Connection Using SSL

The SSL (*Secure Socket Layer*) protocol uses a combination of **public**-key and **symmetric**-key encryption [33]. **Symmetric**-key encryption is much faster than public-key encryption, however

public-key encryption provides better authentication techniques. An SSL session always begins with an exchange of messages called the SSL handshake. The handshake allows the server to authenticate itself to the client by using public-key techniques, and then allows the client and the server to cooperate in the creation of symmetric keys used for rapid encryption, decryption, and tamper detection during the session that follows. Optionally, the handshake also allows the client to authenticate itself to the server. Simple description of SSL handshake follows:

1. The client sends to the server the client's SSL version number, cipher settings, session-specific data, and other information that the server needs to communicate with the client over SSL.

2. The server sends to the client the server's SSL version number, cipher settings, session-specific data, and other information that the client needs to communicate with the server over SSL. The server also sends its own certificate including server's public key.

3. The client uses the information received from server to authenticate the server. If the server cannot be authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the server can be successfully authenticated, the client proceeds to step 4.

4. Using all data generated in the handshake thus far, the client creates the pre-master secret for the session, encrypts it with the server's public key (obtained from the server's certificate), and then sends the encrypted secret to the server.

5. Both the client and the server use the master secret to generate the session keys, which are symmetric keys used to encrypt and decrypt information exchanged during the SSL session and to verify its data-integrity.

6. The client sends a message to the server informing it that future messages from the client will be encrypted with the session key. In addition client sends a separate (encrypted) message indicating that the client portion of the handshake is finished.

7. The server sends a message to the client informing it that future messages from the server will be encrypted with the session key. It addition server sends a separate (encrypted) message indicating that the server portion of the handshake is finished.

8. The SSL handshake is now complete and the session begins. The client and the server use the session keys to encrypt and decrypt the data they send[1] to each other and to validate its integrity.

9. This is the normal operation condition of the secure channel. At any time, due to internal or external stimulus (either automation or user intervention), either side may renegotiate the connection, in which case, the process repeats itself.

6.4.1 Java EE application with SSL

With Sun Java System Application Server, an SSL HTTPS connector by default enabled. To enable SSL in Java EE 5 web application, security constraint have to be specified in application deployment descriptor (`web.xml`). On Figure 6.2 is simple example, how to make security constraint. A user data constraint specifies a transport guarantee (`<transport-guarantee>`). The choices for transport guarantee include following possibilities [35]:

[1] Note that POST requests are in SSL communication deprecated.

- `CONFIDENTIAL` – Sends the data between the client and server in such a manner as to ensure that the true contents are unreadable.
- `INTEGRAL` – Sends the data between the client and the server in such a manner as to ensure that the data are uneditable.
- `NONE` – Requires no transport guarantees.

Authentication mechanism for web module should be in four modes. Authentication mechanism is specified in `auth-method` element in `web.xml`:

- `BASIC` – Uses basic authentication, that is, a user ID and password.
- `DIGEST` – Uses basic authentication (a user ID and password) but sends the password in encrypted format.
- `FORM` – Authenticates by presenting a Web Page for input. Password is send in unencrypted format.
- `CLIENT-CERT` – Authenticates with a client certificate.

Basic authentication sends user ID and passwords over the Internet as text that is base64 encoded, and the target server is not authenticated. If someone can intercept the transmission, the user ID and password information can easily be decoded. Some of these affairs can be diluted when secure transport mechanism such as SSL or security at the network level, such as the IPSEC protocol or VPN strategies is used in conjunction with `BASIC` authentication process.

6.5 Specifying a Security Constraint

Specifying a Security Constraint contains from following parts.

6.5.1 Specifying a authorized users

First of all developer, or application administrator has to specify an authorized user on Application Server. For standard user of the application with rights restrictions on management of a Application Server can be new user specify in `file` realm. This can be made simply with

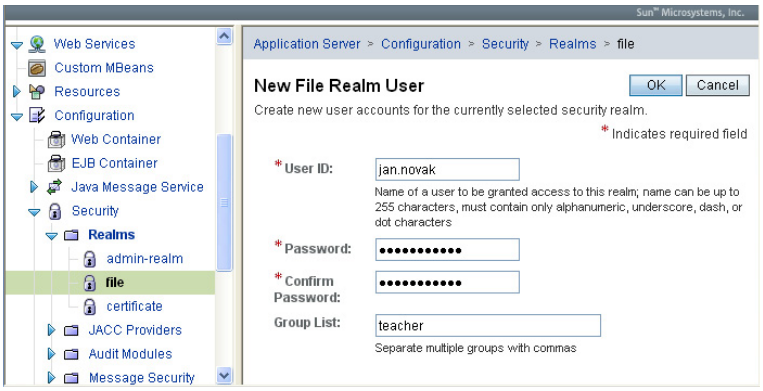


Figure 6.1: Creating user in `file` realm by Admin Console

application console of Application Server. On Figure 6.1 is shown, how to do this task.

6.5.2 Resource constraint in Java EE Application

Now, I have one user *jan.novak* in `teacher` group of `file` realm of Application Server and I would like to make any resource accessible only for users in `teacher` users group. In Java EE are two ways, how to pass this requirement. Through an Annotations or Application DD [34]. Annotations have in most cases some advantages, such as clearness, but for security constraints is in my opinion better to use DD. When both method annotations and DD are used, DD is dominant. When I use some of components created by another developers, I have assurance, that my option is dominant only when I use DD. On Figure 6.2 is any example, how to specify security constraints with DD (`web.xml`) on any type of requests on JSF pages, which are mapped into `/faces/*` path.

```
34      <security-constraint>
35          <display-name>Constraint1</display-name>
36          <web-resource-collection>
37              <web-resource-name>JSF</web-resource-name>
38              <description/>
39              <url-pattern>/*</url-pattern>
40              <url-pattern>/faces/*</url-pattern>
41              <http-method>GET</http-method>
42              <http-method>POST</http-method>
43              <http-method>HEAD</http-method>
44              <http-method>PUT</http-method>
45              <http-method>OPTIONS</http-method>
46              <http-method>TRACE</http-method>
47              <http-method>DELETE</http-method>
48          </web-resource-collection>
49          <auth-constraint>
50              <description/>
51              <role-name>helloUser</role-name>
52              </auth-constraint>
53          <user-data-constraint>
54              <description/>
55              <transport-guarantee>CONFIDENTIAL</transport-guarantee>
56          </user-data-constraint>
57      </security-constraint>
58      <security-role>
59          <description/>
60          <role-name>helloUser</role-name>
61      </security-role>
```

Figure 6.2: `security-constraint` element in DD

`Security-role` element on lines 58–61 in Figure 6.2 specify, which users are known for my web application. In this case it is helloUser. This type of user needn't to correspond with user groups defined in Application Server. For mapping security roles to Application Server principals and groups can be used `security-role-mapping` element in the runtime DD (`sun-web.xml`).

`Auth-constraint` element on lines 49–52 specify which type of user has to be authenticated for this web resource.

`Web-resource-collection` element on lines 36–48 specifies the protected web resource, my own name for this resource and type of requests, which are filtered by this constraint.

`User-data-constraint` on lines 53–56 specify, that data will be transfered over HTTP-SSL. Details about SSL connection was discussed in Section 6.4.

Part II

Virtual Center for Educators of Seniors

Chapter 7

Introduction

While the previous chapters provided short introduction about Java EE technology, this chapter moves on to concrete actual problem domain. Because there are increasing numbers of seniors in the population all around the world, it is very important to provide education for seniors to support their integration into the information society. Virtual Center for Educators of Seniors is a web application, which serves to support education of seniors all over the world. Virtual Center represents an effective way to build an e-community of educators of seniors and to share knowledges and best practices between members.

Main objectives of this part of thesis are to make specification, to design and to implement, test and deliver Virtual Center for Educators of Seniors.

7.1 Skeleton plan

The whole project can be divided into several cascading parts. Some of these proceedings are very often concurrent or recursive as long as all the project members are satisfied. I mean in this case for example two initial activities such as information retrieval and application designing. There is one possible partitioning:

- Basic Study
  - user profiling
  - setting specific goals
  - specifying of use cases and application usability bounds setting
- Analysis
  - other existing systems analysis
  - data analysis (data representation)
  - technology, frameworks, tools
- Design
  - database design
  - component design
  - graphical user interface design
  - testing schedule
  - acceptance test specification

- Implementation
  - self implementation
  - unit tests designing including test processing
- Testing and Installation
  - installation manual
  - installation on customer
  - validation and system testing
  - presentation of the application and documentation delivery
- Support
  - bug fixies
  - by agreement (usability modifications, functionality improvements)
  - technical support by agreement

All these parts are several component parts of project documentation. Common methodology for describing all over the development process is UML (*Unified Modeling Language*). The UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of any system, without an Exception of software system. As well the UML is going to be a standard for software development all over the world for a long time, that was one but not the only reason, why I have made a decision for applying them. For more details about UML or project documentation in general you can read some of references [27, 28] or [7] I have extracted from.

Chapter 8

Basic Study

8.1 User Profiling of the Virtual Center

This chapter is dedicated to gather most of information about modern trends in e-conference scope and e-communities. Before implementation of the application is necessary to be aware of purposes, the application should serve for. In this case, Virtual Center will be service for educators. To make best possible specification of the application I used personas theory to create primary (Figure 8.1), secondary and negative personas (Figure 8.2) representing real user of the application and one user, for that the application is not designed.

Following several sections will be engaged with a number of requirements, I have recognized also through personas.

WANTED
Age: 47
Attitude: extrovert polite patiet calm
Personal goal: go happy life self-pleased
Professional goal: permanent job making career at university
Friends: from university, from sport, from neighbourhood, from internet community
Family: lives with his wife satisfied life has three adult children
Major: teacher, project manager
Computer skills: office software, basic programming
Prefered communication method: eyeball to eyeball, phone (skype), no-SMS
Motivation to use hi-tech devices: make life easier, safe time, contact with friends and family
Travelling: from time to time official journey once a year holiday leave -
Planning trips: allways visiting cities over Europe

Figure 8.1: Primary persona

8.1.1 Support for multimedia

Today most of the conference or community webs try to imitate real discussion between people in digital form. Seniors as well as the educators of seniors are special group of users, which prefer eye-to-eye contact to written form. First requirement is also to design any mechanism providing spoken form of communication. Similarly I have to find any possibility how to share best practices and other study material over spoken form.

For spoken communication between two or multiple members of community exist number of possibilities. Very good experiences I have with Skype software, which enables to communicate with classic phone numbers all over the world too, but above all it is free for use in domain, and has very good support. The calling to another skype user is absolutely free of charge. For all that, Skype technology has own weaknesses, but this is about another story. Another possibility should be another type of VoIP technology, but devices for using this technology are very heterogeneous and control is not always trivial. In addition no another technology has up to date support comparable with skype.

8.1.2 Application Stability

In case the application will be used for creating e-community on a professional level, it is very important the stability of the system. Application must be tested and developed on the same system, on that will be then running. Not only performance of Application and database server, but functionality are very important to achieve stable application. Stability of the system is given by stability of each element of the system. Any unauthorized action from outside the system can essentially decrease system stability, so designing and testing security is also therein very important.

8.1.3 Security requirements

Web applications contain resources, which can be accessed by many users over open network. These resources often go over such open routes, for example as the Internet. Within such environment, a substantial number of web application will require some type of security.

Numbers of web application don't require user registration, but Virtual Center for Educators of Seniors is private application not accessible for everyone. Because the application is running on the server connected to Internet network, there must be implemented some login mechanism and user management to achieve requirement of private accessibility. Simple login mechanism without ciphering sends the user name and password unprotected, so any cipher-mechanism must be also used. Registration will be resolved over an admin user with appropriate access rights (central security management is one of the application requirements). More details about user management will be discussed a bit further. More information not only about Java EE security was discussed in Chapter 6 – Java EE Security (started on page 45).

8.1.4 Internationalization and Localization

Internationalization is the process of preparing an application to support more than one language and data format. Localization is the process of adapting an internationalized application to support a specific region or locale. Examples of locale-dependent information are in addition to messages or user interface labels also character sets and encoding, and date or currency formats. All client user interfaces should be internationalized and localized. In case of web application it is particularly important because of the global nature of the web. Every application with ambitions to be internationalized has to take internationalizing and localizing into account.

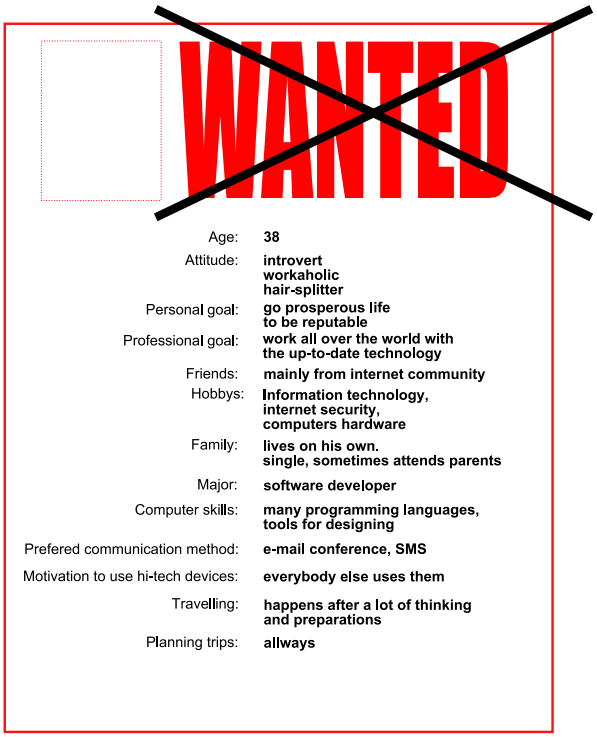

WANTED
Age: 38
Attitude: introvert workaholic hair-splitter
Personal goal: go prosperous life to be reputable
Professional goal: work all over the world with the up-to-date technology
Friends: mainly from internet community
Hobbys: Information technology, internet security, computers hardware
Family: lives on his own. single, sometimes attends parents
Major: software developer
Computer skills: many programming languages, tools for designing
Prefered communication method: e-mail conference, SMS
Motivation to use hi-tech devices: everybody else uses them
Travelling: happens after a lot of thinking and preparations
Planning trips: allways

Figure 8.2: Negative persona

8.2 Setting specific goals

The Virtual Center is dedicated above all for sharing readable and multimedia elements, for creating some closed community as well as for getting know-how in seniors teaching sphere. At first I resume the specific application requirements discussed and described in preceding section:

- community

Group of people with a unity of interest in education or teaching of seniors. Every member will have assign any specific user role according to application privileges.

- conference

Virtual center will represent an instrument for exchange experiences in written or spoken form. Spoken communication will be provided via free Skype client application, whereas for written form will be created embedded discuss.

- tutorials

Virtual center will facilitate to share own created tutorials all over the community. Such tutorials will be in written, spoken or digital movie form. Tutorials will be ordered by branches into the well–arranged list or tree.

- security

Maybe, in the future Virtual Center will contain some private or sensitive data. Application has to be also designed with an advanced security level support. User adding and deleting will be provided by hi–privileged user, or administrator.

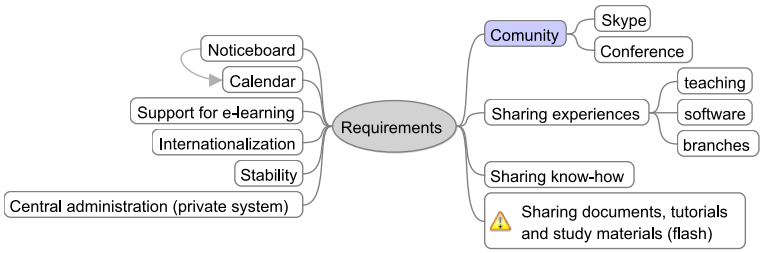


Figure 8.3: Virtual Center requirements

- graphical User Interface

Good usable and utility user interface support for standard users such as admin users. Graphical interface for listing and editing data from database in tabular form.

There are other, not just implemented features, but planned into the future:

- noticeboard with calendar

- e-Learning course support

- open–source and free software promotion

8.3 Use Case

User roles on the Virtual Center are built on few separate privileges. By default there will be set following rights:

- right to vote
- rights for adding / "deleting" discuss
- rights for adding / "deleting" tutorials
- rights for adding / "deleting" users
- admin rights

According to assignment of these several rights to real users, there are four types of users, used in use case diagrams and distinguished by the Virtual Center (obviously, the number of possible user roles is: 2^5, although some of them don't make sense):

	vote	discuss	tutorials	user_mng	admin
unregistered	false	false	false	false	false
registered	true	true	false	false	false
vip	true	true	true	false	false
admin	true	true	true	true	true

Table 8.1: Table of active user roles

Following General Use Case (notice that Use Case with upper cases will stand for a whole diagram whereas use case is only one separate scenario for a single task or goal, in following text) doesn't contain details about stated use cases. Notice, that calling some user by Virtual Center invokes the external Skype application, which has to be at the time already installed.

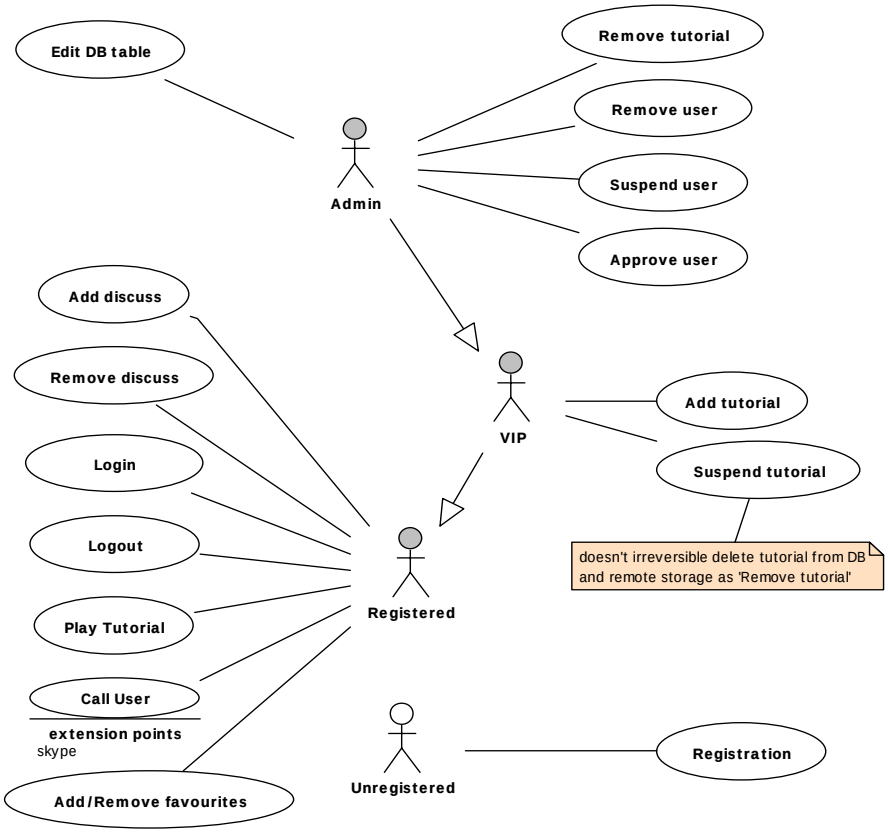


Figure 8.4: General Use Case

8.4 Chosen use cases in higher detail

Users logged to the Virtual Center are able to perform several operations such as search or play tutorial, login, logout and etc., as you can see on Figure 8.4. Following two diagrams (Figure 8.5 and Figure 8.6) show some of referenced use cases in graphical form but more punctually and with some additional dependencies. Use Cases contain only such use cases, which should

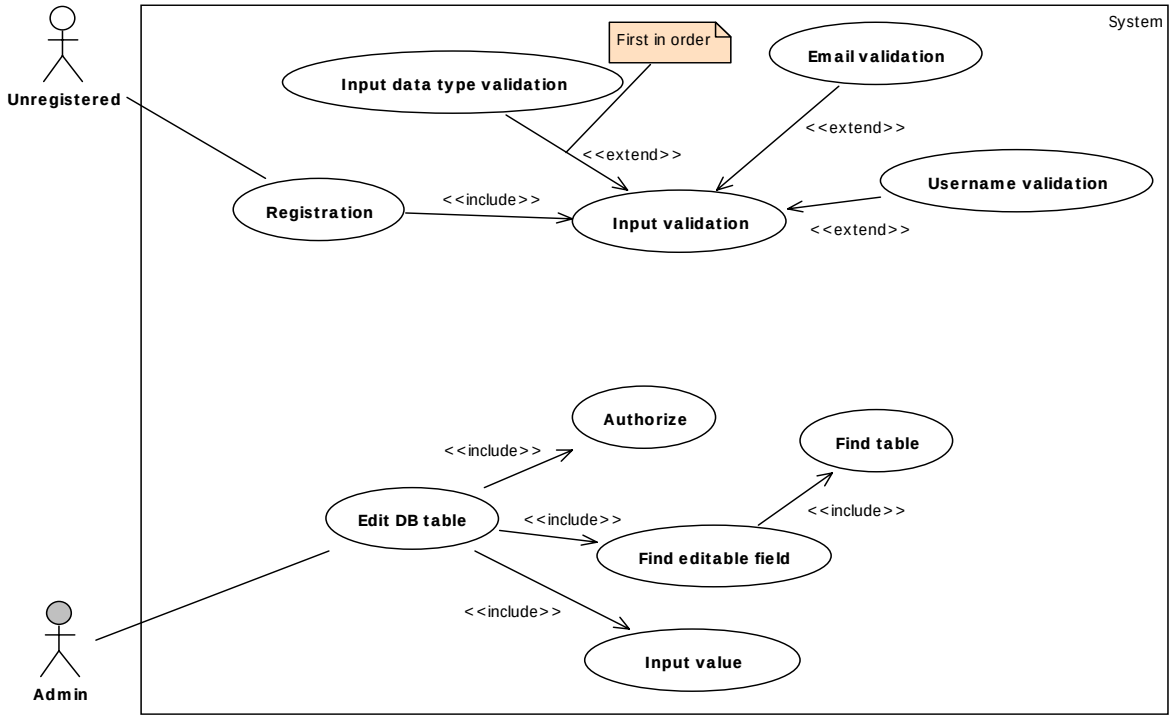


Figure 8.5: Unregistered and Admin user in higher detail

be finished within the scope of first release of Virtual Center. Other functionality similarly referenced in section 8.2 and planned for following releases are not involved.

Notice, that dependencies in Use Cases like `include` or `extend` stereotypes as well as some another elements should be represented by dash line instead of solid one. The impreciseness is caused by unsupporting dashed type of line by exporting in StarUML opensource UML utility, when PDF output is set.

8.5 Usability bounds setting

The Virtual Center is no open system accessible by anyone. Usability is bound on acquirement of valid user name and password. Usually the registration process consists of creating registration request by a new user and sequent approving. In such case admin receives all necessary details of requesting user via fulfilled registration request. Depending on admin's decision, new user account will be approved or not. Anytime every user with privilege for user management can change user status change to suspended, which means that from the time suspended user can't successfully login and use the application.

Application will recognize two independent GUIs. The first one will be for every type of users whereas the second one will be accessible purely by admins and it is supposed for supervision of data tables in DB and checking up stored data. This second GUI's not destined for common usage, but recognition of possible database inconsistences or fault solving mechanisms. Both GUIs will be implemented like a thin client using common web browser.

Finally, the Virtual Center will be supported by Internet Explorer 5 and newer, Mozilla Firefox 1.5.0.9 and newer and Opera 9.0 and newer. Support for another browsers or older releases of enumerated will not be tested during application development process. All the

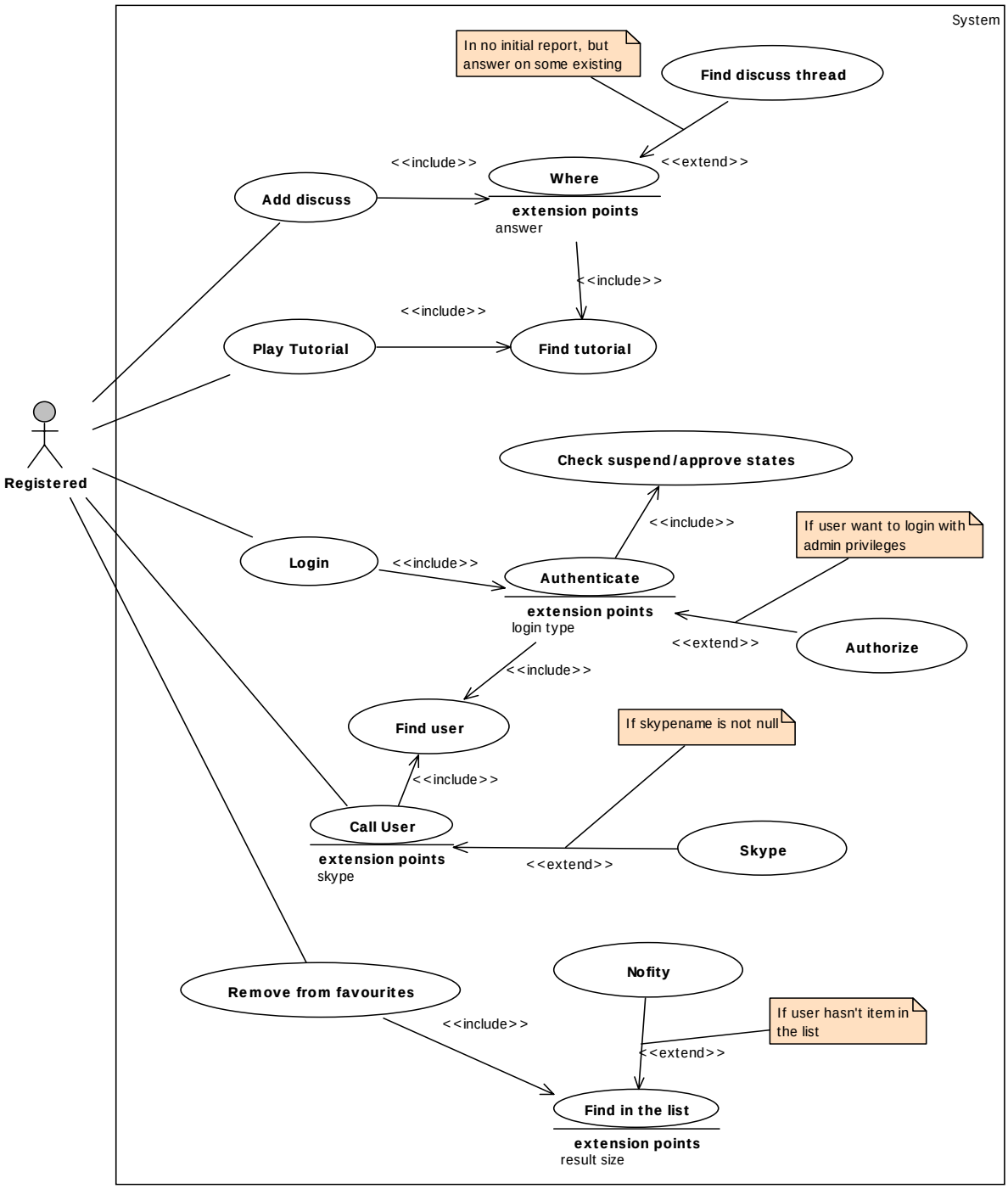


Figure 8.6: Partition of Use Case for registered user

same, the application will generate valid HTML 4.01 code (as possible), should be also well displayed by another clients too.

At the time there is planned no explicit support for using of application on mobile phones or other non-PC devices, although some of technologies described in Part I make it possible. Because of data flow volume in Virtual Center such feature is not found desired today, but regarding to increasing support for Skype and other data technologies, such feature could be released sometime in the future.

Chapter 9

Analysis

9.1 Concurrent systems

Like the TeachUComp[1], also Virtual Center will provide a collection of the most essential skills, every user should know, but free of change for registered users. In addition Virtual Center builds a virtual community (of educators and students), within members will help each other when some problem occurs.

Because the application sketch is very solitary, it is necessary to inspire with other systems, well known for many users. Such system could be e.g e-shops, that provide wares shopping. Virtual Center will be also offering to something, but tutorials instead of wares. The price for wares in virtual center case is free.

9.2 Data analysis

Regarding to preceding chapter and to working knowledge, new data model and data analysis will be designed. Because of possible problems resulting from redesigning of data scheme in the future, database scheme will contain also some tables and structures, which will not to be implemented (eg. vc_elearning, vc_news, vc_portal_messages) in first release of Virtual Center. Such data will not be in question within this section or chapter, but appear in section Database design on page 71.

The entire data analysis is in my case accomplished maybe a bit substandard in database design part. I didn't talk about it, but sometime it is very common that analysis and design parts blend together. Database design has to meet the needs resulting from data analysis. Here are the most important of them in short list: that analysis and design parts blend together. Here are the most important of them in short list:

- user role
  - Admin user is special flag allowing admin GUI access (discussed in section 8.5)
- user
  - User will have user role
  - User will have skypename field
  - User will have list of favourite branches

[1] TeachUComp, Inc. is the only company I have found on internet providing similar type of flash tutorials like virtual Center

  - User will have list of known languages

- branch
  - Branches will be ordered in a tree structure

- discuss
  - Discuss threads will be ordered in a tree structure

- tutorial
  - Tutorial can be voted
  - Tutorial storage will be local for application server view (not remote DB)
  - Tutorial will know its nominal resolution for optimal sharpness of the picture

9.3 Technology, frameworks

Using frameworks, modern IDEs or design patterns make software development much easier and faster then in the past. Because of variety in this branch, it is very important to choose the right tools depending on developed application.

Virtual Center for educators of seniors is a web application using database access. Users will use thin clients via Web browsers as well as administrator. Today there are face to face two strongest technologies, Java EE 5 (earlier J2EE) and .NET. I choose Java Enterprise Technology in version 5 and I'll try to explain my greatest reasons:

1. Java EE is being marketed by an entire industry
2. Java EE is a proven platform, with a few new web services APIs. (Notice, that .NET has quite new reporting services too)
3. Only J2EE lets you deploy web services today !!
4. Existing J2EE code will translate into a J2EE web services system without major rewrites. Not true for Windows DNA code ported to .NET.
5. Java EE is a more advanced programming model, appropriate for well-trained developers who want to build more advanced object models and take advantage of performance features.
6. Java EE gives platform independence.
7. Java EE lets me use any preferred operating system, such as Windows, UNIX, or mainframe. Developers can use the environment, they are most productive in.
8. Java EE lets me use Java, which is better than C# due to market-share and maturity. There are 2.5 million Java developers. IDC predicts this will grow to 4 million by 2003. 78% universities teach Java, and 50% of universities require Java.
9. Netbeans
10. Open and wide Java community
11. Java EE begins to dominate more and more as time goes on and I thing that in Java is future.

9.3.1 Frameworks

From various of Java EE development frameworks, I choose Java Server Faces because it is standard and I prefer standards instead of third parties. From Java EE version 5, JSF is part of technology. Another argument was MyFaces[2] – available components for free usage as well as AJAX (*Asynchronous JavaScript with XML*) components from Java BluePrint Project[3]

As well I though, that Java Persistence API is the rightest way for accessing data from database. Many arguments, as insufficient support from Netbeans IDE or bugs in MySQL make this possibility only theoretical, although very perspective to the future.

Notice that Netbeans 5.5 beta as well as 5.5 final release prefer for web application development `CachedRowSetXImpl` and `CachedRowSetDataProvider` classes for accessing data in database and using of Java Persistence makes in this case of development very uncomfortable. Netbeans inherited most of functionality from Sun Java Studio 2 (known like popular Visual Web Pack feature module), what did Netbeans to framework in itself.

9.3.2 Tools

All the used sw tools are open source, what means that are free for use as same as for development. Usually exists also a community of developers and users what is often the only way to solve possible problems on development.

9.3.2.1 Netbeans IDE

The Netbeans IDE (*Integrated Development Environment*) is up to date the most featured IDE for web development. Another environments usable for creating Java EE application are for example Borland JBuilder[4] or Eclipse[5]. By the work on Virtual center I used Netbeans 5.5 final release with enterprise pack, visual web pack and some of separate modules like Version controll or Sun Java System Application Server support.

NetBeans IDE 5.5
Done loading modules.
NetBeans IDE and NetBeans Platform are based on software from netbeans.org, developed under Common Development and Distribution License (CDDL). For more information visit www.netbeans.org.

Netbeans IDE 5.5 beta

Java EE in Netbeans

Netbeans 5.5 supports the Java EE 5 platform and most notably the EJB 3 and JAX-WS 2.0 specifications. The Java Persistence is not in version 5.5 fully supported (especially in Visual

[2]homepage: `http://myfaces.apache.org/`

[3]homepage: `https://blueprints.dev.java.net/`

[4]Commercial IDE for professional development. Some of developers do not a thing without JBuilder IDE, but in my opinion Eclipse or Netbeans are at least comparable and beyond absolutely free. Homepage of Borland JBuilder is `http://www.borland.com/us/products/jbuilder/index.html`

[5]May be the most popular environment for creating Java application at the moment. Biggest drawback and advantage together in Eclipse is the fact, that whatever developer wants to do, he needs some plug-in for it, that costs very much time spend on browsing over numbers of interested web pages. Home page of Eclipse project is `http://www.eclipse.org/`. Eclipse is as well as Netbeans Open Source project

Web pack) what has to be the gist, why is Java Persistence applied only on parts of the entire Virtual Center application. Netbeans is also advised by Sun Microsystems Inc., that is the most contributor of Java programming language. I think that Netbeans is up to date compared to Eclipse project much more progressive improving environment on the Java programming language IDEs market. As well I think that Netbeans is much more usable for developers compared with Eclipse too (note that Netbeans development team has 25 quality engineers).

Features in Netbeans

- Direct support for Sun Java Application Server PE 9 (Glassfish) – Applications can be deployed into Server direct in Netbeans and then is Server configurable direct from Netbeans nearly in the same way as by Server's configuration console.

- Direct support for numbers of database servers such as MySQL, Java DB, Oracle. Developers can write and run SQL scripts directly from Netbeans IDE. It means that it is possible to connect into selected database and create another new database too. SQL editor knows SQL syntax and include syntax highlighting that makes scripting in Netbeans more attractive. Note that up to date is in Netbeans no code completion's support for SQL file types.

- Direct support for Java Persistence. Netbeans contains many wizards for creating Java Persistence classes.

- Graphical Support for creating XML Deployment Descriptors for Web applications.

- Configurable syntax highlighting for different types of documents, refactoring, code completion, many refactoring features, macros recording, customizable shortcuts and much more.

- Netbeans Visual Web Pack for creating graphical designing of web application

- New support for UML[6] in Netbeans 5.5 Enterprise Pack.

- Integrated solution providing information about runtime behavior of the application calls Netbeans Profiler.

- For more information about Netbeans Features, Addons or for example for seeing many Netbeans tutorials and demos visit home site of Netbeans [26].

9.3.2.2 CVSNT Server

CVSNT is server–side software used to keep a track of changes to files stored on a computer. I used last available version on the date 27.August 2006, version 2.5.03.2382. Client application can be simply Netbeans IDE with installed CVS Versioning System module (version 1.7.22.1.42.1) under Version Control. The home site of the project is `http://www.march-hare.com/cvspro/#free`.

[6]UML(Unified Modeling Language) is in software engineering graphical language for visualization, specification, planing and documentation of development any system. More information about UML you can find in [27]

9.3.2.3 StarUML

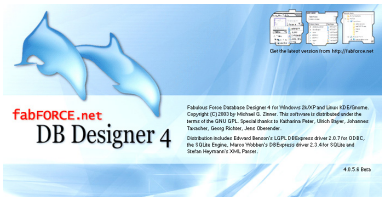
StarUML is an open source project to develop fast, flexible, extensible, featureful and freely-available UML modeling tool. StarUML supports directly UML 2.0 standard. All the UML diagrams such as activity or process diagrams as well as case study from this thesis were made by StarUML tool. StarUML provides simple and powerful plug-in architecture so anyone can develop plug-in modules in COM-compatible languages (C++, Delphi, C#, VB, ...). By design-

StarUML 5.0.2

ing of Virtual Center I used last available build on the date 27.August 2006 (version 5.0.2.1570). StarUML provides features such as Java, C#, C++, PHP code generator (possible next addons through plug-ins), own templates, support for Enterprise JavaBeans .... Unfortunately there is no support for Netbeans IDE and there is the only drawback of StarUML according to me. Home site of StarUML project is `http://staruml.sourceforge.net/en/`

9.3.2.4 DBDesigner

DBDesigner 4 is a free available database design system that integrates database design, modeling, creation and maintenance into a single, seamless environment. DBDesigner 4 is developed



DBDesigner 4.0.5.6 beta

and optimized for the open source MySQL-Database. Together with MySQL server the developer has really very good free tool for developing Java Web Application. Home site of DBDesigner is `http://fabforce.net/dbdesigner4/`.

9.3.2.5 Freemind

Freemind is free mind-mapping application written in Java. Freemind is very practical and effective way to organize ideas. Graphical output is very precious but the biggest advantage I

FreeMind 0.8.0

see in very good user interface, that makes work really fast. Freemind is open source. Home site of Freemind project is `http://freemind.sourceforge.net/wiki/index.php/Main\_Page`.

9.3.2.6 Inkscape

For fundamental graphical design I used Inkscape. Inkscape is very powerful multi-platform vector graphic designer. At work Inkscape is conformable to Adobe Illustrator or CorelDraw.

Inkscape 0.44.1

Some of diagrams in this thesis were made by Inkscape vector tool. I used Inkscape version 0.44.1 available on the date 14.September 2006. The Inkscape's home site is `http://www.inkscape.org/`.

9.3.2.7 Gimp

For cropping images, conversion into raster graphics format, applying any image filters and other work with raster format I used Gimp application. Gimp is free photo software for such

Gimp 2.2.11

tasks as photo retouching, image composition and image authoring. I used gimp only for very elemental working such as cropping images, resizing, compressing and similarly. I used last available build on the date 27.August 2006. The home page of Gimp is `http://www.gimp.org/`.

9.4 Server environment

For my Virtual Center I needed some Application Server for running server-side application. There are numbers of good mostly free and open-source Java Application Servers. From the most popular servers, I choose Java Sun Application Server, because it is very good supported by Netbeans IDE and also it fully supports Java EE 5 standard. Other most popular Java Application servers are Apache Tomcat (`http://tomcat.apache.org/`), JOnAS (`http://jonas.objectweb.org/`) or JBoss (`http://labs.jboss.com/`).

9.4.1 Java Sun Application Server

The Sun Java System Application Server Platform Edition 9.0 is the full name of open-source Java application server developed by Java community like fully compliant implementation of the Java EE 5 platform. I just doesn't mentioned support for the Java EE 5 security model. Application server can be configured for working with the users, groups or roles. There can be as well configured secure HTTP. Some features specific for the Application Server are:

- message security
- single sign-on across all Application Server applications within a single security domain
- programmatic login

Application Server includes a number of Java EE tools that are not part of the Java EE 5 platform but are provided as convenience to the developer. I used only a small fragment of them. There are the most frequently used:

Admin Console – A web-based GUI Application Server administration utility. By developing with Netbeans, it is almost not necessary to use it, because Netbeans IDE contains very good compensation for its. A screenshot of Admin Console is shown on Figure 6.1 on page 49.

asadmin – A command-line configuration utility used for start and stop Application Server and for manage users, resources and applications.

asant – A command-line built-in tool that is an extension of the Ant tool developed by the Apache Software Foundation[7].

verifier – A command-line tool to validate Java EE deployment descriptors.

Sun Java Application Server contains some few components, such as several containers or support for security, transaction and web-services. This work is using last available version of Application Server, version 9.0. Sometimes I met the problem about poor quality of documentation. Except the salaried books I can recommend only [31]. Simple install and user's manual about Sun Java Application Server you can find in Appendix 12.1.

Not only application server but also database server I needed to choose before I could start developing and testing Virtual Center. On the market are also many of database servers. Java can use nearly each of them because of using uniform access to databases over database connectors.

9.4.2 MySQL

For exemplary purposes I used MySQL database server in version 5.0.24a and MySQL Java connector version 5.0.3. For accessing database from web application were:

1. Started MySQL server (Install and User's tutorial will be in Appendix 12.1).
2. Populated database in database server
3. Connected database with Netbeans IDE
4. Included MySQL connector library into Virtual Center

Similar proceeding can be used by another database servers too. MySQL has very good user's and developer's documentation at home site: `http://www.mysql.org/doc/refman/5.0/en/index.html`. Alternative database servers are Oracle, PostgreSQL, Derby database distributed with Application Server and many others.

[7] Homepage of Apache Software Foundation is `http://www.apache.org`

Chapter 10

Design

Design part of sw development life–cycle should start after finishing previous parts, because all the requirements including possible extendability of an application, used technology, frameworks etc. should be already known and design has to take all these apects into account.

Within my project of Virtual Center, I see just here the greatest imminence of the whole developent process. Explanation comes through. Although I have made very intimate analysis of possible user from last chapter and all the requirements were determined according to him, it is probable that as early as during multiyear usage of the application, there will be rising new and new requirements. It is also one good reason for using component-based framework for development, because finished components should be very simply reused.

10.1 Database design

For demonstrative purposes will be used free MySQL database version 5, because of very good experiences with MySQL 3 together with J2SE application development, open source philosophy and very good documentation built on wiki engine.

The whole ER model including some up to date unused entities like vc_elearning or vc_news is on Figure 10.1 on page 72 in crows foot notation. The complete report which describes the same database scheme in text notation including data types of fields, matching primary and foreign keys, as well as obligation of some fields, default values if set and other details, is attached in Appendix A.

Thanks to the unified SQL language and Java Connector Architecture, it is very convenient to migrate to another database such as Oracle, Firebird, PostgreSQL or another when MySQL doesn't correspond with growing requirements in the future.

10.2 Component design

The whole Virtual Center consist of JSP pages, that contain JSF components. The architecture of components nor list of used components are interesting at the time. I said, that Visual Web Pack is as matter of fact a framework in itself. I'll also describe structure of application, how could be designed with this development tool. Firstly, we have to familiarize with chapters 1, 2, 3 and 4 from part I of this thesis.

Web client – The Web client is represented by any supported web browser. User via web client accesses the server. According to `web.xml` deployment descriptor, start page is opened.

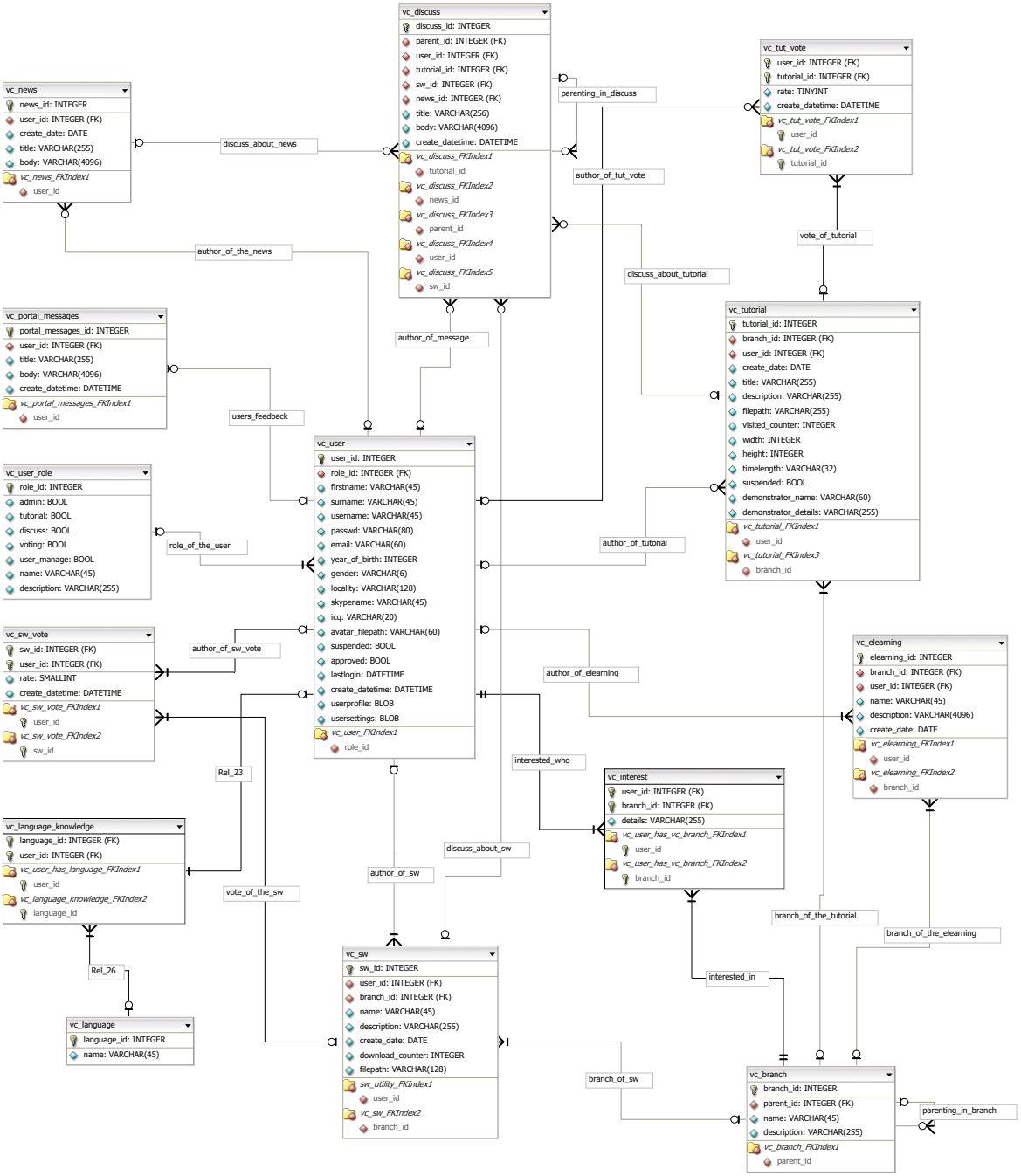


Figure 10.1: ER model

Glassfish server 9.01 – Sun Java System Application Server is an opensource server–side utility (known as Glassfish) listening on specified port and providing servlet container service among others.

web.xml – Deployment Descriptor file configuring the virtual center application

Servlet – Every JSP page is compiled into the servlet class according to the JSP life cycle (page 19).

JavaBeans – JavaBean objects are divided into four types according to scopes, which they belong to. Depending on which scope the JavaBean belongs, Glassfish manages its life-cycle. JavaBeans are configured by the `managed-beans.xml` descriptor file. JSF recognizes four types of scopes: Application, Session, Page, Request. The only Application JavaBean could be shared by multiple relations.

CachedRowSet – In a nutshell: "A CachedRowSet object is a container for rows of data that caches its rows in memory and makes possible to operate without always being connected to its data source. Further, it is a JavaBeans™ component and is scrollable, updateable, and serializable." [37]

DB server MySQL – Well documented open source database server (page 69).

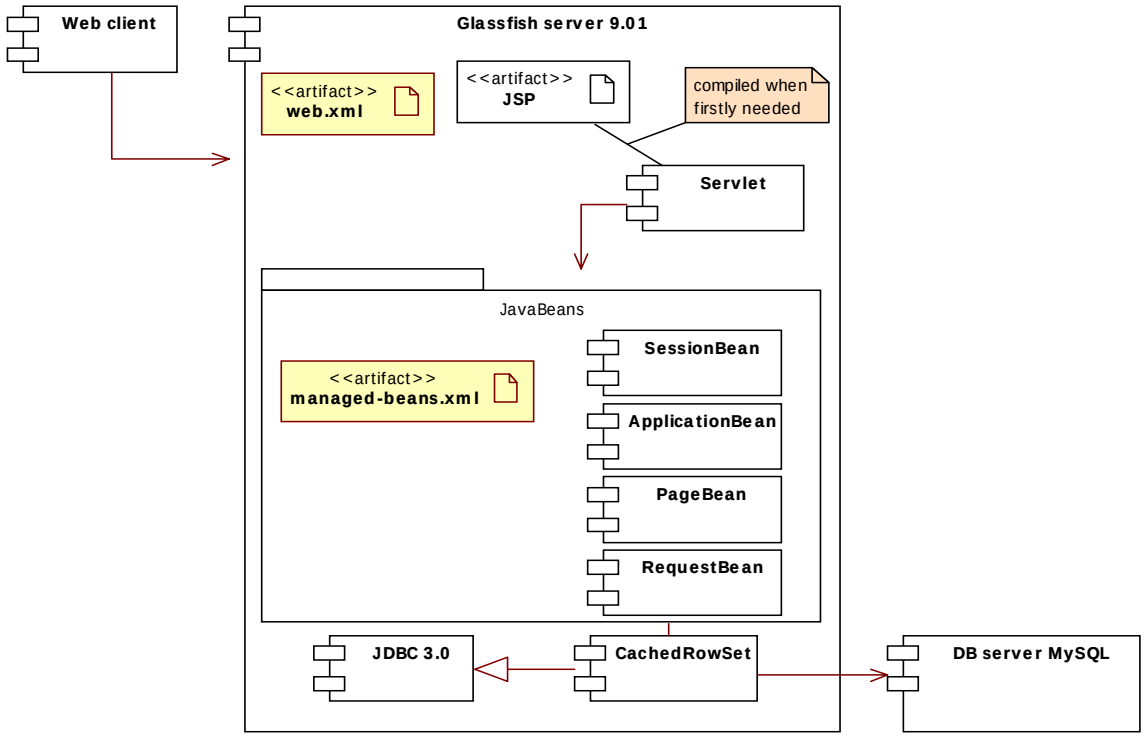


Figure 10.2: General Web application component diagram

10.2.1 Page structure

This section describes page structure and simple navigation over the Virtual Center (Figure 10.3). Diagram shows only actual planned functionality. When new page is added, no changes

are needed, developer only append some necessary navigation rules, especially disconnect for example.

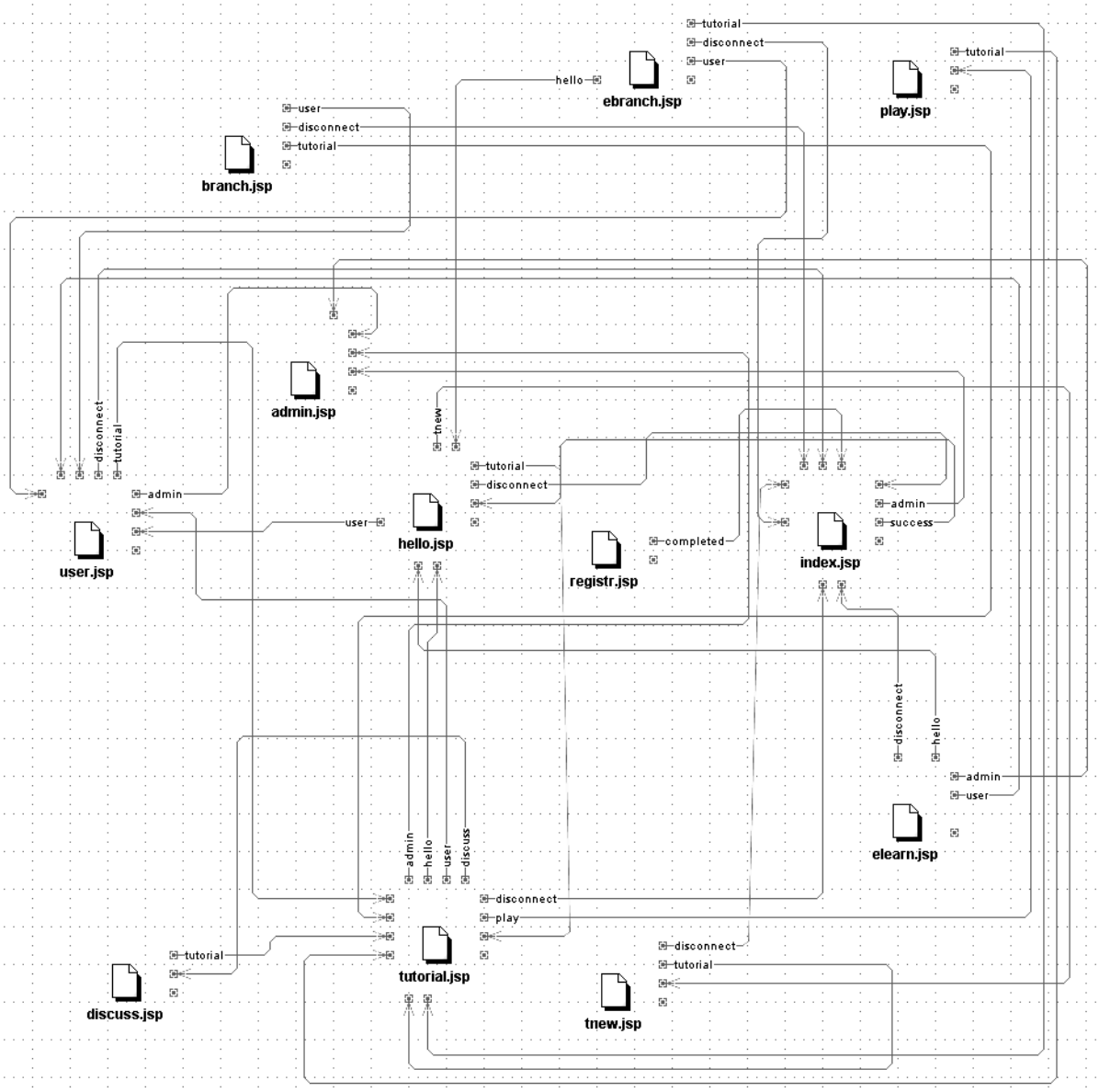


Figure 10.3: Page navigation diagram

10.2.2 Pages description

This section describes behaviors of particular forms (pages) from previous section and for the most interested of them will be also designed their action diagrams like a template for implementation. Forms with elementary behavior or analogous to some another form will not be designed here.

hello

Hello is the start page for basic user, where the user is automatically redirected after successfully login (for administrators, *admin* page is the starting one). Main functionalities of *hello* page are to present firstly the most important information about state of Virtual Center and secondly

changes since user connected last time. *Hello* page makes all the Virtual Center functionality accessible for logged user (depending on user privileges naturally). Last but not least the *hello* page displays personal preferences for actual logged user, because the application makes possible some customizations. On Figure 10.4 is shown mechanism of login, the primary way for accessing the *page*.

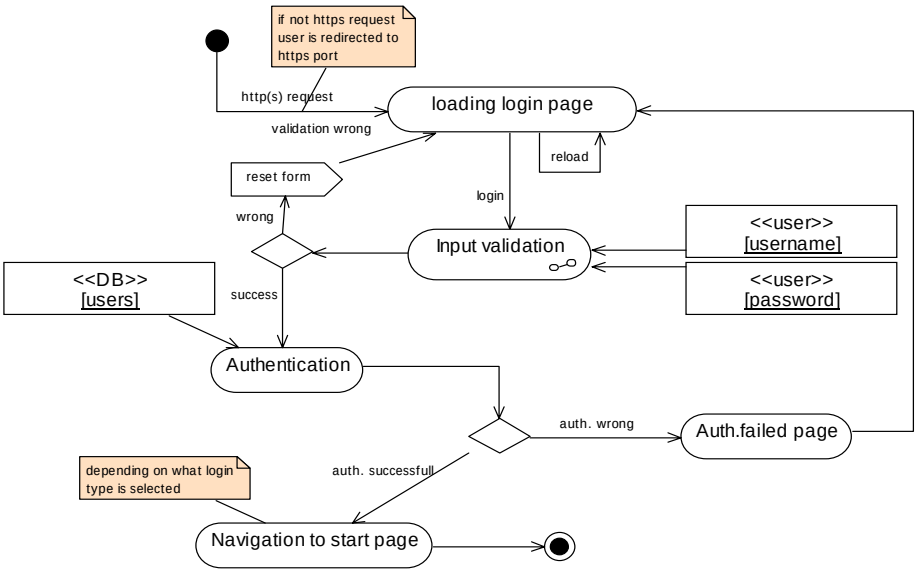


Figure 10.4: Login activities scheme

admin

Analogous to *hello* page for basic user, admin user has *admin* page. Administrator can use advanced functionality not allowed for basic user like user management, persistent delete tutorials or another.

branch

Branch form (page) is predeterminated for showing tutorials related to selected branch. Navigation to the page can be alternative. First possibility is the typical JSF navigation by configuration file (`navigation.xml`. Because there is not possibility to send information about selected branch via this type of navigation, this information has to be stored within some shared object. In my case, I'll use `SessionBean` object, because there could arise some problems when using RequestBean. Punctual reasons will be trashed out in following chapter of this thesis. The second possibility is navigation via url hyperlink including id parameter of selected branch. Figure 10.5 shows both possibilities of navigation of the *branch* page.

ebranch

Analogous to *branch*, *ebranch* displays all related e-learning courses according to the branch selected in the menu. Navigation and functionality are very analogous with *branch* page too.

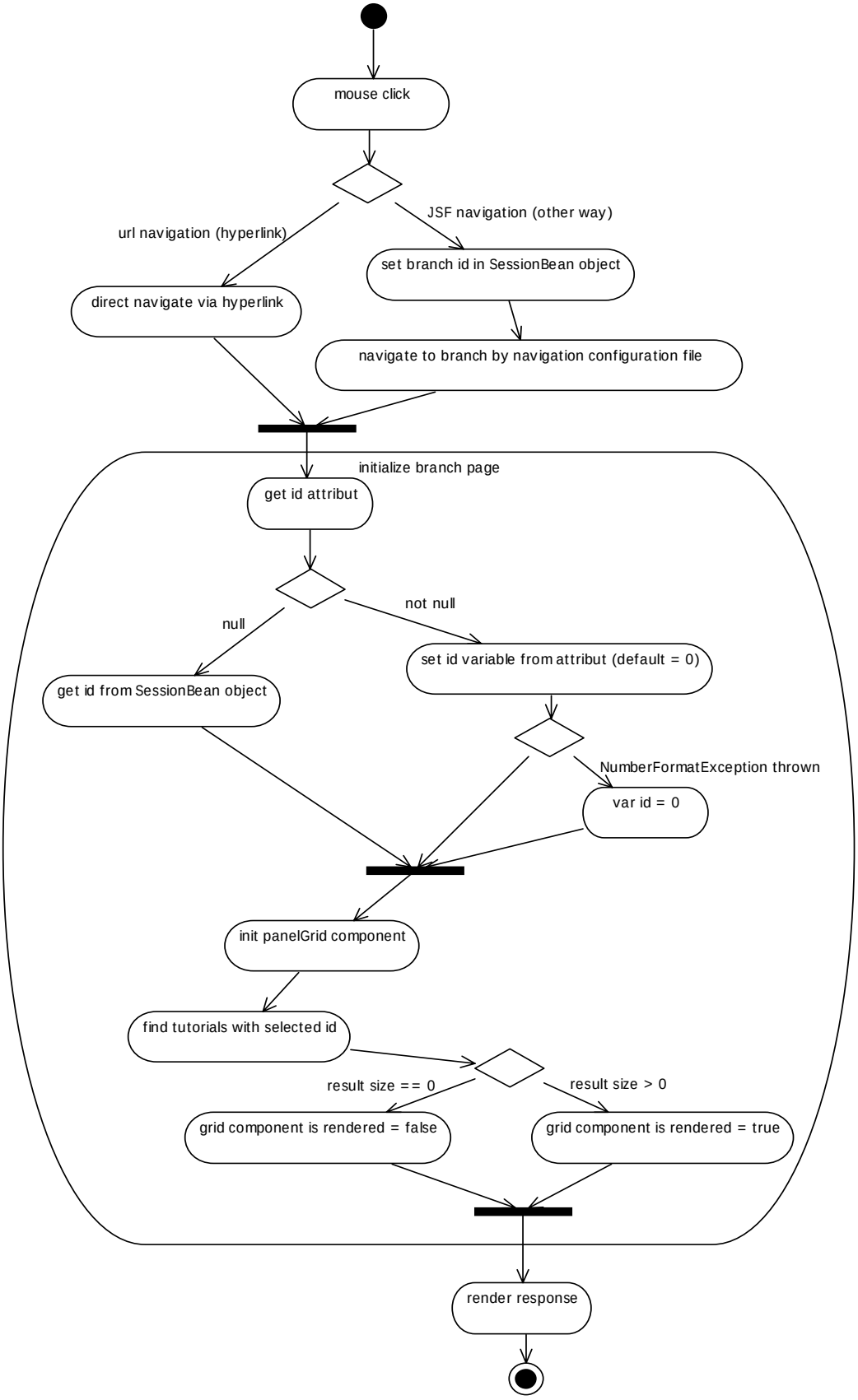

mouse click
JSF navigation (other way)
url navigation (hyperlink)
set branch id in SessionBean object
direct navigate via hyperlink
navigate to branch by navigation configuration file
initialize branch page
get id attribut
not null
null
set id variable from attribut (default = 0)
get id from SessionBean object
NumberFormatException thrown
var id = 0
init panelGrid component
find tutorials with selected id
result size == 0
result size > 0
grid component is rendered = false
grid component is rendered = true
render response

Figure 10.5: Diagram shows how is navigated *branch* page

tutorial

Tutorial page shows the most of known details about selected tutorial. *Tutorial* page is accessible via many different ways, eg. from main menu, user favourites menu, *branch* page, *user* page .... User can according to his rights add or delete discuss related with tutorial as well as can watch the tutorial. Use Case on Figure is a roughcast of main functionalities of the *tutorial* page.

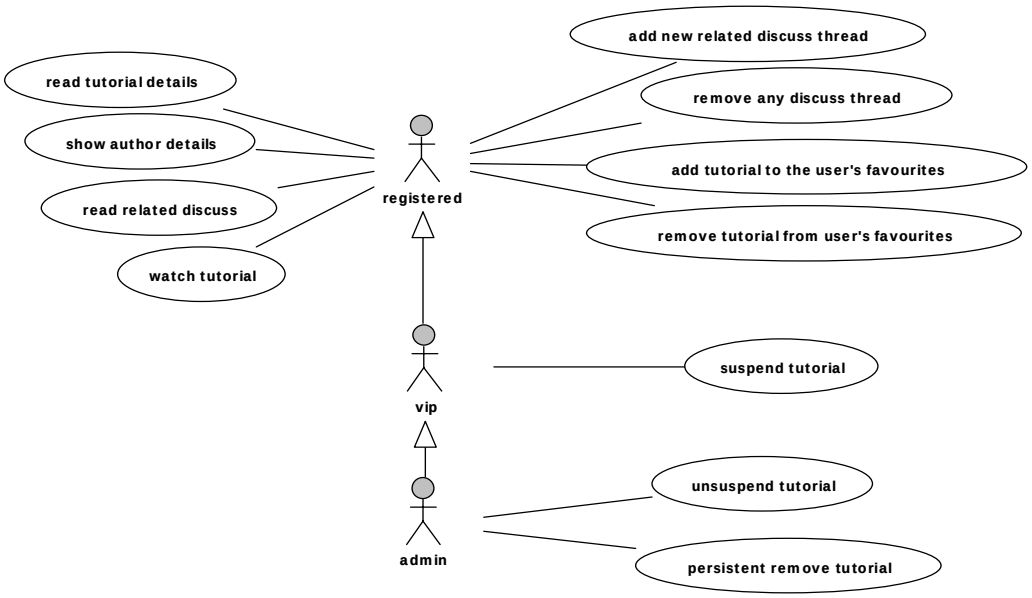


Figure 10.6: Use Case of *tutorial* page

play

This page is predeterminated for watching tutorials. The only additional functionality for *play* page is an interface for voting tutorial. User can vote from 1 to 5, where 5 is the best. When user's vote will be entered, user's view will be redirected to the *tutorial* detail page.

user

Analogous with *tutorial* page, *user* page shows details about selected user. Naturally, there is no possibility to "play user", in its place are contact user by skype or contact user by email client.

elearn

Analogous with the *tutorial* or the *user* pages, details about selected e–learning course are selected by the *elearn* page. At the time the only gathered details are according to the ER scheme (Figure 10.1) title, description or creation date. Support for e–learning courses is planned to be improved in the future, but due to the Virtual Center doesn't want to supply systems that have to be already exist, such functionality has to be very punctually analyzed before some revolutionary improvement would be performed in this direction.

discuss

The *discuss* page contains a simple form with two text fields or text areas supposed for inserting title and body of actually creating new discuss thread. The only additional data necessary for creating new discuss are author's id and parent discuss for creating relationship between parent and child in discuss table hierarchy. The author's id is already stored in the session scope (because he is actually successfully logged and all the information about them are also available in special session scope). Id of the parent discuss is conveyed in the same way as by accessing *branch* or another pages.

tnew

The *tnew* form is supposed for inserting new tutorials to the virtual center. One `swf` file mustn't to be bigger then 30MB, otherwise the creating tutorial will be unsuccessful (application setting).

10.3 Graphical user interface

Usability and graphical user design are together the most important issues in software (application) development. The Virtual Center is implemented to provide many user-friend features such as customization, quick access to the most frequently or best rated elements etc.

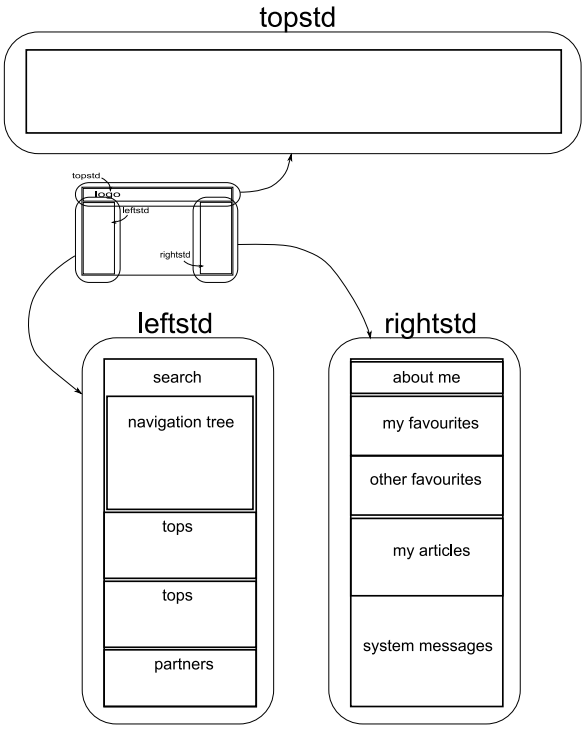


Figure 10.7: Graphical user interface partitioning (most of pages)

Very good practices for building good navigable and ergonomic graphical user interface under web application development are to keep some conventions. Today developers more often disuse dynamic navigation menu, because of different visibility in different web clients. Another philosophy could be to identify, which web client application user uses and then can be rendered different navigation menu according to the client type. Such solution is not systematic and it

is very difficult to maintain such application for all users using different web browsers in the time, apart from JavaScript support necessity. Other convention is to divide the whole gui into separate parts (divs) according to their functionality. Sure I mustn't forget on requirement to write valid HTML code.

The other practice and one of the most important principles in multi-tier applications user interface designing is to fully separate user interface design from the business logic. JSF technology enable more then dividing of these parts one from another, actually it is possible to create absolutely different renderers for different devices opening the same application. Developer only needs to implement such renderers. I talked about renderers in section 10.5 talking about UI Components and Component tree in JSF.

10.3.1 Page framing

I decided to divide typical page's scheme into four separate blocks.

topstd (Top standard block) – Most of the navigation will be concentrated in other blocks. Topstd panel will also rather serve to inform user about his identification, eventually can perform some unique operations like e.g. accessing starting page from this block.

leftstd (Left standard block) – The most of navigabilities in the application are divided into Left and Right standard block. The main functionalities accessible from leftstd are according to Figure 10.7 searching and navigating. There is variety of possibilities, how to access information, that user actually wants to see. Firstly he can use search engine. At the time, the searching is built on the knowledge of key value of the element, which user want to actually see (tutorial, discuss). Other way to access element, which user is actually interested in is to use the navigation tree or a pregenerated list from the offer of best rated.

rightstd (Right standard block) – Instead of leftstd, the rightstd is partly customizable and makes the navigation really interested. User can assign the most important elements according to his privileges into prepared checkboxes and then he can simply navigate them without the necessity of looking them repeatedly. Naturally when user is logged out, the customize settings are persistent. Under these lists of favourites user can simply navigate to the tutorials, he was discussed about them recently from "my articles" field in rightstd block. Last in the rightstd is a place intended for printing system messages if some error in application occurs or some irreversible operation is performed.

The middle layout – The middle layout block is the main part of displayed page. Other three parts are included into this main part as three separate frames. The middle layout is predeterminated for displaying most of the information, that user is actually interested in. This part of displayed page is for every other page different.

10.3.2 Colors

Very important part of graphical design is also color adjustment. I choose from modest colors between blue and green for the bulk of whole graphical design. The important parts I colored by red or another well contrasting color according to the neighborhood of the part. The obligatory fields are marked by red asterisk mark.

10.3.3 Ordering

According to my own experience, when the page contains more elements like tutorials for example, I prefer to order them into a grid (e.g. like some e–shops does[1]). This is a bit problem to make a grid in strict JSF component structure of the page. (It is not possible to use elements like `<h:foreach...` because life-cycle of JSF components differs from simple HTML or JSP components.) One solution is to create a `GridLayout` (or any other) with one column and into each of the rows to bind one row from table with three columns.

Very welcomed feature should be selecting superior branches, when user shows for example branch "Visual Web Pack" in tree-menu "Netbeans", user get hyperlinks to Netbeans. Analogous, when for example user would select 36SI3, user get sorted *ČVUT > FEL > Department of CSE > 36SI3*, where ČVUT, FEL and Department of CSE are active hyperlinks (breadcrumbs) for example.
Examples of GUI of Virtual Center like monitor screenshots are appended in Appendix B.

10.4 Testing

One of the main goals of this thesis is making robust secure application. Firstly we have to say, that it is nearly impossible to develop well tested, absolutely bug-free application in semi-annual or annual sw project. I know by experience, that 50–80% of all the usability of functional bugs in application are discovered during first one or two years of active usage according to the type of project, although the application was thoroughly tested before their release[2]. Because my technical knowledges about Java EE and used tools were on the start of this project very limited (Java EE released in the middle of last year and Netbeans 5.5 together with VWP as late as last October) and the documentations of both are very limited at the time, the potential risk of nonexecution the project was very high. I also have to design for the project such testing and project schedule, which makes such risk as low as possible.

10.4.1 Generally about test processing

Main objective of tests I scheduled is to cover required application quality all over the development process. The developer, who is responsible for testing in Java EE development team, is tester. These tests in this chapter are limited on covering the quality of the code. I'm not engaged in problematics of analysis or design of application here.

10.4.2 Testing strategy

1. ***Unit tests*** – these tests are focused on application module verification (like JavaBeans, codes behind, classes ...). According to description of the methods there are most important routes in module are tested here. Single parts of unit tests are white box as well as black box testing. For unit testing will be used JUnit module in Netbeans, which is able to generate some elementary unit tests actually by itself.

2. ***Integration tests*** – integration up to down. The beginning is on the main module and the progress leads down (e.g. Breadth-First-Search). The main module is "driver", "stubs" substitute all the subordinated modules. In time stubs are replaced by real modules.

[1] See Alzasoft (`http://www.alza.cz`) or OxyOnLine(`http://www.oxyonline.cz/`)

[2] Approximately two years I'm regular reader and contributor in the `nbuser` community, and I know that every day are discovered several new issues and bugs in Netbeans IDE although they have 25 quallity engineers in the team.

When new module is integrated, the system is tested again. A synonym for such testing is *regress testing*.

3. ***Regress testing*** – checks if an error occurs when new module is added. Previous tests are reprocessed. The error detected by this tests has to be called displaced error.

4. ***Validation testing*** – Validation testing follows integration tests, verifies fulfilment of "reasonable expectations" of customer – validation criteria. Validation tests are performed by black–box methods. Alpha tests as well as beta tests are both accomplished. Alpha tests are realized by customer in the environment of supplier, whereas the beta test will be executed by customer in service conditions. Summary of results achieved in course of alpha part is available on page 96.

5. ***System testing*** – A set of tests, that verify the entire system including hardware, people...

Security testing – This test tests immunity of the system against an unauthorized attack. Tester in "hacker's" role try to penetrate into the application. If he has enough time and resources he succeeds. The objective of developers is the biggest costs of the penetration.

Recovery testing – System testing which verifies, if all the faults were corrected in a proper time. For auto error handling reinitialization, checkpoint mechanism, data recovery and restarting are evaluated.

Stress testing – System testing on base of extreme load in light of quantity, frequency or content. Tester measures the time until the application breakdown.

10.4.2.1 Test reports

Accent is put on comparison between requirements and quality of the results. How are requirements realized and how much the real application differs from them. Important approaches about rectification of invalid parts will be designed in the scope of this report. After corrections all affected tests will be revised. In case of extensive corrections, new set of tests have to be scheduled.

All the test reports will contain date, test fulfillment characteristics and headline of testing results. In conclusion will be these results compared with requirements and component will be evaluated according to them.

10.5 Acceptance test

Acceptance test is build on user requirements and on application functionalities.

Acceptance tests are black box system tests. Each acceptance test represents some expected result from the system. Customers are responsible for verifying the correctness of the acceptance tests and reviewing test scores to decide which failed tests are of highest priority. Acceptance tests are also used as regression tests prior to a production release.

Acceptance tests should be automated so they can be run often. The acceptance test score is published to the team. It is the team's responsibility to schedule time each iteration to fix any failed tests.

10.5.1 Conditions of acceptance tests

The Virtual Center application will run on the server machine, and the connected database runs on the same computer, or on another one. For testing modules is necessary to ensure access rights to the application module and to the operation system on the computer, where the application runs.

In the database is implicitly created admin account with unlimited access all over the application's components and functionalities. Admin account can also create new admins or another types of users. Testing consistence of the database will be provided by set of SQL scripts, which will produce tables of data and possible inconsistence. In the application will be put emphasis on the security of the web application too. The greatest part of acceptance tests will be apply to presentation logic and business logic of the application (if the application does in the appropriate way, what user expect).

10.5.2 Documentation requirements

Integral part of the application is the necessary documentation. It consist of these separate parts:

- Install and uninstall user manuals (digital form)
- Technical documentation for the application (Part II from this thesis)
- Source codes (digital form)

10.5.3 Functionality requirements

Functionality is depending on the user role of the signed–in user. A collective requirements for all tested cases are:

- web browser supporting HTTP over SSL
- internet (LAN) connectivity (or installed Virtual Center on local machine).

Action	**Registration request approving**
Description	Admin user (or any other user with user management right) can approve registration request to enable user account. Without this action new user can't use the Virtual Center application..
Requirements	• user account with admin or user management rights • submitted registration request from new user (not approved)
Progress	1. Login with admin user. 2. Show all not approved registration requests. 3. Choose one user from the list. 4. Press `approve` button. 5. Try to login with new user.
Possible output	**[OK]** – New user is successfully logged in. **[OK]** – No not-approved user in the list. You have to create new registration request firstly. **[Error]** – System message about "User approved" was cought, but the new user can't login since the system message about "not approved.." is thrown. Fatal error. Approve operation was not performed. Suggestion: an error within the `login()` method. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection cann be established. Approve action wasn't performed. No suggestion.

Table 10.1: Registration request approving

Action	**User suspending**
Description	Admin user can suspend any other user to disable his user account. Suspended user account can be unsuspend again. Only privileged users with admin or user management rights are able to perform this functionality.
Requirements	• user account with admin or user management rights • user, who should be suspended has to exist and mustn't to be already suspended.
Progress	1. Login with admin user 2. Show all users. 3. Choose one user from the list. 4. Press `suspend` button. 5. Try to login with suspended user.
Possible output	**[OK]** – Suspended user can't login now. System message about "suspending" of the user is caught. **[Error]** – Suspended user logged successfully. Suspending operation has not be performed although an successful message was caught. Suggestion: problem in method enganged with suspending the user. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection cannot be established. User wasn't suspended.

Table 10.2: User suspending

Action	**User unsuspending**
Description	Admin users can unsuspend any suspended user. Valid (unsuspended) user account can be suspend again. Only privileged users with admin or user management rights are able to perform this functionality.
Requirements	identical with Table 10.2 except for suspended instead of unsuspended
Progress	identical with Table 10.2 except for unsuspend instead of suspend
Possible output	**[OK]** – Unsuspended user successfully logged again. **[Error]** – Unsuspended user can't login successfully although a system message about "user unsuspended" was caught. Suspending operation wasn't performed. Suggestion: an error should be within the method enganged with user unsuspending. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection cannot be established. Unsuspend action wasn't performed. Admin can try to repeat the action.

Table 10.3: User unsuspending

Action	**Switching user roles**
Description	Admin user or any user with user management roles can switch user roles for another users in VC. Operation is reversible, but only for privileged user.
Requirements	• user account with admin or user management rights • any other user from Virtual Center.
Progress	1. Login with admin user 2. Show all users. 3. Choose one user from the list. 4. Choose from possible user roles and press `Set` button. 5. Choose from the list of users the same user as in 3. 6. Check actual user role.
Possible output	**[OK]** – Actual user role matches my selection. **[Error]** – Actual user role doesn't match to the value, I've chosen. User role changing operation has not to be performed. Suggestion: an error in method which gather user role from the checkbox within the `user` page. **[Error]** – System message about "A DataProviderException thrown" is caught and printed. Connection cannot be established.

Table 10.4: Switching user roles

Action	**Tutorial deleting**
Description	Admin users can delete any tutorial regardless of suspended value. Tutorial file is removed from specified storage as well the tutorial record is removed from database. All related discuss for this tutorial is also removed from database according to cascade delete set in MySQL.
Requirements	• user account with admin rights • any existing tutorial
Progress	1. Find tutorial in list of suspended tutorials or alternatively and navigate to its details. 2. Remember tutorial id number in up right corner of tutorial detail form. 3. Press "red" `delete` button on tutorial detail page. 4. Try to find the deleted tutorial by known id from 2 using search engine.
Possible output	**[OK]** – Tutorial cannot be found. **[Error]** – You have found deleted tutorial. Details are displayed and tutorial file size is valid number. **[Error]** – You have found deleted tutorial. Details are displayed, but tutorial's file size is "file not found". Tutorial was deleted from local storage, but removing tutorial record from database was unsuccessful. **[Error]** – System message about "a DataProviderException thrown" is caught. Connection with database can't be established. Tutorial is probably deleted from local storage (this operation precedes deleting from database). **[Error]** – System message about "Tutorial can't be deleted from local storage" is caught. An `IOException` thrown during deleting tutorial. Tutorial wasn't deleted from local storage, nor from database. May be, that tutorial is in use at the time when deleting.

Table 10.5: Tutorial deleting

Action	**Login (admin)**
Description	Admin and basic users have both a bit different graphical user interface to improve usability of the application.
Requirements	• user account with admin rights
Progress	1. Navigate to login page. 2. Switch combobox to admin type, fill the user name and password and press `Login` button
Possible output	**[OK]** – You are redirected to the admin's start page. **[OK]** – You are not redirected to the admin's start page and system message about "you are suspended user at the time" is caught. You are suspended user at the time. **[OK]** – System message about "you are not admin" is caught. Application doesn't know, that you are admin user. The user who is trying to login into admin UI, has no admin rights. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection with database can't be established. Try again.

Table 10.6: Login (admin)

Action	**Login (user)**
Description	Admin and basic users have both a bit different graphical user interface to improve usability of the application.
Requirements	• user account with admin rights
Progress	1. Navigate to login page. 2. Switch combobox to user type, fill the user name and password and press `Login` button
Possible output	**[OK]** – You are redirected to the user's start page. **[OK]** – You are not redirected to the user's start page and system message about "you are suspended user at the time" is caught. You are suspended user at the time. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection with database can't be established. Try again.

Table 10.7: Login (user)

Action	**Tutorial suspending**
Description	Simple user is not able to delete tutorials from the Virtual Center, with tutorial rights he is able to suspend them. Suspended tutorials are not visible for any simple users. Admin users can delete or unsuspend any suspended tutorial for persistent delete or returning visibility to the tutorial.
Requirements	• user account with vip rights • any existing unsuspended tutorial
Progress	1. Find tutorial in Virtual Center. 2. Remember tutorial id number in up right corner of tutorial detail form. 3. Press "gray" `Delete` button on tutorial detail page. 4. Try to find the deleted tutorial by known id from 2. Use search engine.
Possible output	**[OK]** – Tutorial is suspended. As a result, the tutorial details can't be accessed any more. **[Error]** – System message about "Tutorial was suspended" was catched, but details of tutorial can be accessed. Suggestion: The Virtual Center ignores its invisibility or tutorial's suspended value wasn't changed. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection with database can't be established. The Virtual Center is consistent. Try to find the tutorial again. If you find him, repeat the progress from 2. No suggestion.

Table 10.8: Tutorial suspending

Action	**Tutorial unsuspending**
Description	Suspended tutorial is visible only for admin user. Admin user can such tutorial permanently delete, or unsuspend to retrieve him visibility for everyone.
Requirements	• user account with admin rights • any suspended tutorial
Progress	1. Find tutorial in list of suspended tutorials and navigate to its details. 2. Remember tutorial id number in up right corner of tutorial detail form. 3. Press "gray" `Unsuspend` button on tutorial detail page. 4. Try to find the deleted tutorial by known id from 2 using search engine.
Possible output	**[OK]** – Tutorial was found, details were displayed correctly. **[Error]** – No tutorial was found. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection with database can't be established. Tutorial is still suspended. Try to repeat the whole progress.

Table 10.9: Tutorial unsuspending

Action	**Adding new tutorial**
Description	Every user with tutorial right (admin or vip) can create own tutorial and import them to the Virtual Center.
Requirements	• user account with at least vip rights • existing tutorial in x-shockwave-flash data format
Progress *5. wait...	1. Go to start page of the Virtual Center (logged as user). 2. Click the Add button in the top right corner of the middle layout. 3. Fill the form and choose your data file with file chooser. (File mustn't be bigger then 30MB, othercase the operation fails – the value is adjustable by server administrator in `web.xml`) 4. Press the `Store` button.
Possible output	**[OK]** – Tutorial detail page is displayed. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection with database can't be established. Tutorial is not stored in the Virtual Center. **[Error]** – System message about "A IOException thrown" is caught. Virtual Center can't write tutorial to the specified storage. Suggestion: Maybe, there is not enough space on the storage.

Table 10.10: Adding new tutorial

Action	**Adding e–learning course**
Description	Every user with at least (vip) rights can insert information about existing e–learning courses, that is visible for another users of Virtual Center.
Requirements	• user account with at least vip rights • knowledge about any e-learning course
Progress	1. Go to start page of the Virtual Center (logged as user). 2. Choose `e-learnings` above branches tree in leftstd block. 3. Choose the branch, where you can insert new e-learning course information. 4. Go to the bottom of the middle layout, where is the label `New tutorial`. 5. Fill the form. 6. Press the `Create` button. 7. Navigate to the `e-learnings` and choose the same branch as last time.
Possible output	**[OK]** – On the page is list of e-learnings related to the selected branch, and on the top of the list is your e-learning. New e-learning course information are stored in the Virtual Center. **[Error]** – The e-learning you have inserted recently is not in the list. There is probably some bug in the method saving the e-learning. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection with database can't be established. E-learning course information are not stored in the Virtual Center.

Table 10.11: Adding e-learning course

Action	**Deleting information about e-learning course**
Description	Every user with at least (vip) rights can delete information about existing e-learning courses, that he stored earlier.
Requirements *Progress*	• user account with at least vip rights 1. From leftstd ▷ e-learnings ▷ *some branch* choose any e-learning, that actually logged user created. 2. Press `Delete` button on the top of the page. 3. Navigate to the `e-learnings` and choose the same branch as last time.
Possible output	**[OK]** – On the page is list of e-learnings (if any) related to the selected branch, and your e-learning is not in the list. Your e-learning course was removed from the Virtual Center. **[Error]** – The e-learning you have deleted recently is still in the list. Suggestion: There is probably some bug in the method deleting the e-learning. **[Error]** – System message about "A DataProviderException thrown" is caught. Connection with database can't be established. E-learning course information weren't deleted.

Table 10.12: Deleting information about e-learning course

Action	**Create new discuss thread**
Action	**Remove discuss thread**
Action	**Reply existing discuss**
Action	**Add user / tutorial to the favourites**
Action	**Remove user to the favourites (button / hyperlink)**
Action	**Remove tutorial to the favourites (button / hyperlink)**
Action	**Call user with skype**
Action	**Email to the user**
Action	**Find tutorial by id**
Action	**Find discuss by id**
Action	**Add registration request**
Action	**Navigate tutorial from the list of favourites**
Action	**Navigate user from the list of favourites**

Table 10.13: Addional tested functionalities

Chapter 11

Implementation

In consideration of possible improvements of Virtual Center in the future, I choose Java EE technology because in my sketchy analysis of two strongest technologies .NET and Java™, Java turned out to be a better choice. Just at the moment Java offers in my opinion more features, better community, very good IDE Netbeans with Visual Web Pack and better application architecture. In addition there is "new" application server Sun Java System Application Server 9 resulting from previous versions of commercial Java Application Server up to version 8 and developed as well under opensource project – named Glassfish.

Very good facility of Java EE and Glassfish server is the possibility to debug web applications. No other technology provides such instrument for web developers.

Because Java technology is not bound on one concrete database product and because I have had very good experiences with using Java together with MySQL on standard desktop applications, I choose for demonstrative purposes MySQL 5 database server for this web project too.

11.1 Self implementation

Following few sections will be engaged with describing some of my own solutions. For enclosed parts of codes, I'll keep syntax highlighting to make these codes as much readable for readers as possible.

11.1.1 Some Netbeans conventions in Java EE development

Every IDE as well as Netbeans has its own conventions, which should be learned before developer starts to use them for software development. Such conventions in Netbeans I would sectionalize by abstraction level. Some of the most important JSF components of the Virtual Center are summarized in Table 11.1.

Database handling

For binding data from database into Java EE application are in Netbeans Java EE project used by default `CachedRowSetXImpl` and `CachedRowSetDataProvider` object. RowSet objects are by default stored in SessionBean1 class, that is default session scope object in the project.

While the `CachedRowSetXImpl` does the real work of inserting data into a database table, the `CachedRowSetDataProvider` provides an easier coding alternative. Rather than coding directly to the RowSet, can be used the DataProvider methods to access the

scope	scope's class	usage of the class
Application beans	ApplicationBean1	– not used –
Session beans	SessionBean1	data shared all over the session. As well I used this session for some parameter passing, because the request beans are very limited (see next section, part about the Page navigation).
	UserProfile	information about actually logged user, such as user's access rights, user's username, last user's login, user preferences... Password is not stored within this session because of possible security risk.
Pages beans	<<page bean>>	class representation of component tree of the page. Most important parts of all the page beans are: `init()` method initializing UI components. `init()` method is called as the first when the page is navigated to, or when some event occurs, for example. `prerender()` method called before render response phase is performed. This is good place for redirecting an request for example. `destroy()` callback method that is called after rendering is completed for this request, if `init()` was called (regardless of whether or not this was the page that was actually rendered). `Destroy()` method is good place for releasing of reserved resources. Page beans contain only data needed for one response life–cycle. `init()` and `destroy()` are typical for all Java Beans. Session and Request Beans has in addition `activate()` and `passivate()`. `prerender()` and `preprocess()` callback methods are in Netbeans environment typical only for Pages Beans.

Table 11.1: Implemented JSF source components

RowSet. Whether or not DataProvider is used, however, the CachedRowSet has to be created. Keep in mind that if a database table is dropped on to a page, the Netbeans automatically creates a `CachedRowSetXImpl` (in the `SessionBean1 by default`) and a `CachedRowSetDataProvider` (in the page itself – "code behind") for that table. It is the `CachedRowSetXImpl`'s command property, that is the SELECT statement, that provides the key to the database INSERT operation.

The table INSERT command generated by the IDE includes all the columns that are in the `CachedRowSetXImpl`'s command property. For example, suppose the RowSet's command property is "*SELECT ALL elearning_id, branch_id, user_id, name, description FROM vc_elearning*". The INSERT command for this RowSet then becomes "*INSERT INTO mytable (elearning_id, branch_id, user_id, name, description) VALUES (?,?,?,?,?)*".

Code example on Figure 11.1 illustrates DataProvider methods how can be simple used for inserting a row into a data source. Firstly the `canAppendRow()` method followed by `appendRow()` method are used and if it is possible to append row, then an empty row is appended as result to the `CachedRowSetDataProvider`. Then call `setCursorRow()` to set the DataProvider's cursor to the new row's location. The same `setValue()` method is used to set the different column values, regardless of the column types. Finally, `commitChanges()` method is called to committing the changes to the data source. A simple `catch` statement is needed to catch some errors that might occur.

Instead of appending a row, the `CachedRowSetDataProvider.insertRow()` method can be used to insert a new row before a specified row. If so, it is necessary to know the `RowKey` value for the row before that the new row should be inserted and to pass that value as a parameter to the `insertRow` method.

```
public String createElearning() {
    try {
        if( elearningDP.canAppendRow() == true ) {
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd");
            Date now = new Date(System.currentTimeMillis());

            //append row if possible
            RowKey appendRow = elearningDP.appendRow();
            elearningDP.setCursorRow( appendRow );

            //set the new data values
            elearningDP.setValue("vc_elearning.branch_id", id);
            elearningDP.setValue("vc_elearning.user_id", getUserProfile().getUser_id());
            elearningDP.setValue("vc_elearning.name", newItem[0]); //nameField.getText
            elearningDP.setValue("vc_elearning.description", newItem[1]);
            elearningDP.setValue("vc_elearning.create_date", sdf.format(now));

            elearningDP.commitChanges();
            elearningDP.refresh();

            info("new tutorial added");
            return "hello";
        } else {
            error("can't add vc_elearning row");
            return null;
        }
    } catch (DataProviderException ex) {
        log("cannot add new elearning because of dataProviderException thrown");
        ex.printStackTrace();
        info("cannot add new elearning course (DataProviderException)");
        return null;
    }
}
```

Figure 11.1: Code example: *createElearning()* method

Page navigation

According to diagram on Figure 10.3, JSF supports two types of navigation. Direct (URL) and indirect (navigation rules in navigation configuration file). When client is using cookies, there is no problem with associating user with session, because the session id is stored in the cookie. When user client doesn't support cookies, or want to disable them, the application developer has to look for any alternative way. Such possibility is sending session's id via each url request and response within the session.

When the direct navigation type (url) is used, the developer has to ensure that every url was encoded by `encodeURL()` method. This method adds to the encoded url, session's id. JSF components call this method automatically, but when some servlet or scriptlet in JSP is used, `encodeURL()` method has to be called explicitly.

When the indirect navigation type (navigation rule) is used, the developer has to be very careful not to miss possible attributes sent by request bean object. An original request is lost and a new one is created. The only possibility how to traverse this is to use some superior scope to request like for example session scope. The page navigation diagram you can see on Figure 10.3 on page 74.

11.1.2 Login, logout and data security

All passwords of users are in the Virtual Center encrypted with MD5 algorithm. I decided to implement my own MD5 algorithm to become the same result as `md5()` function in SQL for MySQL. The most important method from `MD5Passwd` class is `encodeHeslo()`, which is in the code example on Figure 11.2. Login page is encrypted according to the deployment descriptor configuration file (DD), password is also always dispatched in encrypted form and can't be misused by third party. There is a very important mechanism, which forbids render

```
public String encodeHeslo(String heslo, String method) {
    java.security.MessageDigest mdAlgorithm = null;
    try {
        mdAlgorithm = MessageDigest.getInstance(method);
    } catch (NoSuchAlgorithmException ex) {
        ex.printStackTrace();
    }
    if (mdAlgorithm != null) {
        mdAlgorithm.update(heslo.getBytes());

        byte[] digest = mdAlgorithm.digest();
        StringBuffer hexString = new StringBuffer();

        for (int i = 0; i < digest.length; i++) {
            heslo = Integer.toHexString(0xFF & digest[i]);

            if (heslo.length() < 2) {
                heslo = "0" + heslo;
            }

            hexString.append(heslo);
        }

        return hexString.toString();
    }
    return null;
}
```

Figure 11.2: Code example: *encodeHeslo()* method

response phase, if user is not logged in. If user "fakes" for example the URL in web browser, he is automatically redirected to the login page. (See Figure, there is my own solution of redirecting in this case). For logout mechanism I also need only to invalidate actual session objects to lose all information about user and new login is necessary before next using of virtual center.

Security constraint are quite large field in Java EE and it is possible to set all constraints entirely in DD, but the functionality of such solution is very limited. User is able or not to access the resource, we can't have different appearance of the page for different user roles, we have to implement completely different page for different user roles. That was the main reason, why I decided for referenced solution.

Upload binary file (tutorial)

There are several possibilities for uploading binary file, although "The most JSF" solution is using `uploadFile` component from AJAX JSF components. Because I have had a pretty issues in using this upload component, there is a skeleton solution of the method by creating new tutorial. Firstly the file path from upload component has to be obtained. I wanted to store all tutorials into the domain's folder, a relative folder path has to be also translated into real path for the operating system. Maybe, that file path already exist, I append also filename with an counter. The rest of the `createButton_action()` method creates new record in database `vc_tutorial` table, that is very similar to `createElearning()` on Figure 11.1.

11.1.3 Issues

- MyFaces components don't support Glassfish server, but the support is planned into the future. This information is gathered from official web page (`http://myfaces.apache.org/compatibility.html`) of MyFaces project up to date 12. January 2007.
- The support for JPA in Netbeans Visual Web pack was very poor up to December 2006. Other side `CachedRowSetXImpl` usable from JDBC 3.0 is great alternative.

```
/**
 * <p>Callback method that is called just before rendering takes place.
 * This method will <strong>only</strong> be called for the page that
 * will actually be rendered (and not, for example, on a page that
 * handled a postback and then navigated to a different page).  Customize
 * this method to allocate resources that will be required for rendering
 * this page.</p>
 */
public void prerender() {
    if(getUserProfile().isLogged() == false || getUserProfile().isSuspended() == true) {
        info("You are not logged in!");
        String theURL = "index.jsp";

        FacesContext faces = FacesContext.getCurrentInstance();
        ExternalContext context = faces.getExternalContext();
        HttpServletResponse response = (HttpServletResponse)context.getResponse();
        try {
            log("redirecting");
            response.sendRedirect(theURL);
            log("response lost");
            faces.responseComplete();
        } catch (IOException ioex) {
            log("IOEX " + ioex.getMessage() + " JK.");
        } catch (IllegalStateException isex) {
            log("ISEX " + isex.getMessage() + " JK.");
        }
    }
}
```

Figure 11.3: Code example: *prerender()* method

```
public String logoutButton_action() {
    ExternalContext econtext = FacesContext.getCurrentInstance().getExternalContext();
    Object session = econtext.getSession(false);
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy.MM.dd hh:mm");
    log(sdf.format(new java.util.Date(System.currentTimeMillis())) +
            "username: " + getUserProfile().getUsername() +
            " is removing old session");

    if( session != null ) {
        log("usersettings backup");

        getUserProfile().persistentUserSettings();

        ((HttpSession)session).invalidate();
        System.out.println("Session was just invalidated, follow redirect " +
                "to login page");
    }
    return "disconnect";
}
```

Figure 11.4: Code example: *logoutButton_action()* method

- MySQL connectors version 5 and newer have problems in conjunctions with JPA. Numbers of exceptions are thrown including an exception about *"Unable to retrieve EntityManagerFactory...*[1]

11.2 Unit tests designing and processing

I used JUnit module in Netbeans for creating JUnit tests. JUnit tests are automatically generated and the only task is to change "todo" part of each testing method. JUnit tests use variables: `instance`, `result` and `expResult`. Firstly an instance of the class is created afterwards a tested method is performed and `result` is compared with `expResult` in `assertEquals` method, which asserts that two objects are equal. If they are not, an `AssertionFailedError` is thrown.

[1]Notice that I haven't try to use the newest MySQL Connector/Net 5.0.3 GA from 04. January 2007, although according to the specification in release changes the bug #22425 wasn't fixed.

```
public String createButton_action() {
    UploadedFile uploadedFile = fileUpload.getUploadedFile();
    String uploadedFileName = uploadedFile.getOriginalName();

    int index = uploadedFileName.lastIndexOf('/');
    String justFileName;
    if ( index >= 0) {
        justFileName = uploadedFileName.substring( index + 1 );
    } else {
        // Try backslash
        index = uploadedFileName.lastIndexOf('\\');
        if (index >= 0) {
            justFileName = uploadedFileName.substring( index + 1 );
        } else {
            // No forward or back slashes
            justFileName = uploadedFileName;
        }
    }

    String uploadedFileType = uploadedFile.getContentType();

    int lastdot = justFileName.lastIndexOf(".");
    String prefix = new String(new SimpleDateFormat("yyMMdd").format(
            new Date(System.currentTimeMillis())));
    String message = new String();
    if ( uploadedFileType.equals("application/x-shockwave-flash") ) {
        CachedRowSetDataProvider tutorialDP = new CachedRowSetDataProvider();
        try {
            ServletContext theApplicationsServletContext =
                    (ServletContext) this.getExternalContext().getContext();
            String realPath =
                    theApplicationsServletContext.getRealPath
                    ("/data/swf");
            int counter = 0;
            File file = new File(realPath + File.separatorChar + prefix + "_" +
                    justFileName);
            while ( file.exists() == true ) {
                file = new File(realPath + File.separatorChar + prefix + "_" +
                        counter++ + "_" + justFileName);
            }
            message = "new tutorial was created was created";

            uploadedFile.write(file);

            try {
                // there is a set of commands creating new row into vc_tutorial
            } catch (DataProviderException ex) {

                ex.printStackTrace();
            }

        } catch (Exception ex) {
            message = new String("Cannot upload file: " + justFileName + "\n" +
                    ex.getMessage());
            return null;
        } finally {
            // release resources
            tutorialDP.close();
        }
    } else {
        // not possible to upload not swf file
        message = "The only supported file type at the time is x-shockwave-flash.";
        error(message);
        return null;
    }
    return "tutorial";
}
```

Figure 11.5: Code example: *createButton_action()* method

Chapter 12

Testing and installation

This chapter is engaged with the final parts of the Virtual Center's development. Installation and deploying of web applications are generally much more complicated then by a simple desktop application and so, I'll try to describe the whole process with more details.

12.1 Installation manual

I will divide the whole installation process into separate concuring parts. Notice that this installation manual is punctually applicable only for the Virtual Center's installation on MS Windows XP. If you are using a different operation system, the installation process should also a bit different.

- Install JDK 5 (if not already done)
  1. Run */app/jdk-1_5_0_06-windows-i586-p.exe* from enclosed CDROM
  2. Follow the installation wizard
- Install SJSAS 9 (if not already done)
  1. Run */app/sjsas_pe-9_0_01-windows.exe* from enclosed CDROM
  2. Follow the installation wizard
  3. Choose your password for accessing admin console
- Install MySQL 5 (if not already done)
  1. Run */app/mysql-essential-5.0.24a-win32.msi* from enclosed CDROM
  2. Follow the installation wizard
  3. Run MySQL instance Config Wizard (if not started automatically)
  4. Step by step choose:
     `Reconfigure Instance`
     `Detailed Configuration`
     `Developer Machine`
     `Multifunctional Database`
     Choose any path, where you want to store data from database
     `Decision Support`
     Check `Enable TCP/IP` and `Strict Mode`. Choose the port `3306`
     `Manual Selected Default Character Set` – `utf8`
     `Install As Service` and `Include bin directory in PATH` – both
     Set password `'toor'` for the root user, don't create anonymous account
     `Execute`

- Install MySQL GUI Tools
  1. Run */app/mysql-gui-tools-5.0-r3-win32.msi* from enclosed CDROM
  2. Follow the installation wizard
- (Restart computer when needed)
- Start MySQL server service
- Start Sun Java System Application Server 9
- Run `Query Browser` from `MySQL gui tools`
- Create new database
  1. Fill the form:
     Server host: localhost Port: 3306
     Username: root
     Password: toor
     Default schema: vc
  2. Confirm, that you want to create new schema 'vc'
- Create data scheme
  1. `File` – `Open Script`
  2. Choose */vc/web/database/createscript.sql* from enclosed CDROM
  3. `Execute`
- Insert example data into database
  1. `File` – `Open Script`
  2. Choose */vc/web/database/loadscript.sql* from enclosed CDROM
  3. `Execute`
- Create new Connection Pool in the Application Server
  1. Run `Start` > `Sun Microsystems` > `Admin Console`
  2. Login with 'admin' username and your admin password
  3. Choose `Create new JDBC Connection Pool` from the main frame
  4. Fill the form:
     Name: `mysqlPool`
     Resource Type: `javax.sql.DataSource`
     Database Vendor: `mysql`
  5. Next
  6. Fill the form:
     Transaction Isolation: `read-uncommitted`
     Check Isolation Level: `Guaranteed`
  7. Delete all predefined properties

8. Add following properties:
   `serverName` : `localhost`
   `poolNumber` : `3306`
   `Password` : `toor`
   `User` : `root`
   `databaseName` : `vc`

- Create new JDBC Resources

  1. In Admin Console, left tree menu choose: `Application Server` > `Resources` > `JDBC` > `JDBC Resources`
  2. Press `New` button
  3. Fill the form:
     JNDI Name: `jdbc/dataSource`
     Pool Name: `mysqlPool` Enabled: checked

- Deploy the Virtual Center application

  1. From left tree menu choose: `Application Server` > Applications > Web Applications
  2. Press `Deploy` button
  3. Choose the file: */vc/dist/vc.war* from enclosed CDROM
  4. `Next`
  5. Check: `Precompile` : enabled

- Logout Admin Console
- Start the Virtual Center

  1. Start your favourite web browser
  2. type url: `https://localhost:8181/vc/faces/index.jsp`

- Additional information
- According to */vc/web/database/loadscript.sql* the Virtual Center knows following users
  - username: jarda.kortus, password: heslo, user role: admin
  - username: martin.kortus, password: mheslo, user role: registered
  - username: helenka.holikova, password: hheslo, user role: vip

This tutorial is available in dynamical shockwave flash form, in /demo/demos.html folder on enclosed CDROM.

12.2 Validation test

According to designed acceptance test (page 81) and requirements of the user, a validation test over the functionality of VC was performed. Table 12.1 gives a summary about alpha validation of VC including list of all elements that were tested and short notations about fulfilment of each element, if passed or not.

✓	**Create new registration**
✓	**Approve new user**
✓	**(Un)Suspend user**
✓	**Switch user roles**
✓	**Delete tutorial (admin)**
✓	**Delete tutorial (user)**
✓	**Login (admin)**
✓	**Login (user)**
✓	**Unsuspend tutorial**
✓	**Add new tutorial**
✓	**Add elearning course**
✓	**Delete Elearning course**
✓	**Create new discuss thread**
✓	**Remove discuss thread**
✓	**Reply existing discuss**
✓	**Add user / tutorial to the favourites**
✓	**Remove user to the favourites (button / hyperlink)**
✓	**Remove tutorial to the favourites (button / hyperlink)**
✓	**Call user with skype**
✓	**Email to the user**
✓	**Search tutorial by id**
✓	**Search discuss by id**
✓	**Add registration request**
✓	**Navigate tutorial from the list of favourites**
✓	**Navigate user from the list of favourites**

Table 12.1: Summary of validation test results

Part III

Conclusion

Chapter 13

Conclusion

This section concludes the thesis with a discussion of the proposed concepts and achieved results. The main goals of the thesis were study of Java EE application architecture model together with modern Java EE development process and implementation of the server–side application for improving education of seniors.

The first part of this whole thesis is enganged in single steps and principles of Java EE development process. Foremost the Java EE technology and application architecture are described.

In the middle of the thesis, there is one chapter about Java Persistence API. Java Persistence API is not entirely supported by development tools and user interface components at the time and that was the main reason, why I have finally decided to use another technology (Chapter 11.1.1 about `CachedRowSets` starts on page 87) for Virtual Center's database handling. Java Persistence API is a new feature in Java EE version 5, therefore this chapter stayed in the thesis although another technology was choosed for the database handling of the Virtual Center application.

The power of object relational mapping is shown on web application for table manamement of my Virtual Center's database. In demos (*/demos/index.html*) there is a short demostration of `vc_JPA` application developed wholly by Java Persistence API in Netbeans. `Vc_JPA` application is no UI Component using simple web application for database table management used during the development of Virtual Center.

The Java EE is a quite large field and it is also not possible to cover every element of Java EE architecture or development process within this thesis. An intention was to give essential overview about architecture and development principles in Java EE for the reader. Simultaneously, I thing that the thesis contains description of new elements in Java EE version 5 and in addition, I tried to give enough arguments, why I thing, that Java EE is better than Microsoft .NET (page 64) in field of web development. For more informations about Java, Java EE, web development, multitier architecture, object relational mapping, Netbeans or other described technologies and tools, you can consult with some of referenced bibliography starting on the page 115.

Second part of the thesis presents Java EE technology and development tools on creation really existing system. The Virtual Center for educators of seniors (VC), which is the main deliverable of the EU SENNET project, should provide a stable platform for building e-community. VC should be able to gather knowledges and best practices, helping to improve the professional qualifications and skills of its individual members. Although the main target group of the project are educators of seniors, it will in the end, just the seniors themselves will

benefit from the outcomes of the project. In this way, more and more seniors will be integrated into the information society.

Although VC covers all requirements, that were established before the project started, there are many things enhancing both the conceptual and programmatical solutions provided in this thesis. There are the most important improvement which could extend functionality and usability of VC, whereas some of them were planned already in a frame of this project to the future, but were not finished:

- Sharing of useful software utilities similarly like sharing tutorials
- Support for other types of tutorials (like movies, static presentations. . .)
- More customizations (e.g. resizing of played tutorial)
- Author of thread or admin should be the only users able to delete discuss thread
- Support to add new branch to the branche's tree dynamically
- Progress bar during upload of a long file
- Calendar of events (Conferences, Workshops)

Appendices

Appendix A

Database report

vc_tutorial

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
tutorial_id	**INTEGER**	PK	NN	UNSIGNED			AI
branch_id	INTEGER		NN	UNSIGNED			
user_id	INTEGER		NN	UNSIGNED			
create_date	DATE						
title	VARCHAR(255)						
description	VARCHAR(255)						
filepath	VARCHAR(255)		NN				
visited_counter	INTEGER			UNSIGNED			
width	INTEGER			UNSIGNED			
height	INTEGER			UNSIGNED			
timelength	VARCHAR(32)						
suspended	BOOL						
demonstrator_name	VARCHAR(60)				null		
demonstrator_details	VARCHAR(255)				null		

IndexName	IndexType	Columns
PRIMARY	PRIMARY	tutorial_id
vc_tutorial_FKIndex1	Index	user_id
vc_tutorial_FKIndex3	Index	branch_id

vc_interest

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
user_id	**INTEGER**	PK	NN	UNSIGNED			
branch_id	**INTEGER**	PK	NN	UNSIGNED			
details	VARCHAR(255)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	user_id branch_id
vc_user_has_vc_branch_FKIndex1	Index	user_id
vc_user_has_vc_branch_FKIndex2	Index	branch_id

vc_user_role

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
role_id	**INTEGER**	PK	NN	UNSIGNED			AI
admin	BOOL		NN				
tutorial	BOOL		NN				
discuss	BOOL		NN				
voting	BOOL		NN				
user_manage	BOOL		NN				
name	VARCHAR(45)		NN				
description	VARCHAR(255)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	role_id

vc_tut_vote

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
user_id	**INTEGER**	PK	NN	UNSIGNED			
tutorial_id	**INTEGER**	PK	NN	UNSIGNED			
rate	TINYINT			UNSIGNED			
create_datetime	DATETIME						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	user_id tutorial_id
vc_tut_vote_FKIndex1	Index	user_id
vc_tut_vote_FKIndex2	Index	tutorial_id

vc_language_knowledge

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
language_id	**INTEGER**	PK	NN	UNSIGNED			
user_id	**INTEGER**	PK	NN	UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	language_id user_id
vc_user_has_language_FKIndex1	Index	user_id
vc_language_knowledge_FKIndex2	Index	language_id

vc_language

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
language_id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(45)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	language_id

vc_language

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
language_id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(45)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	language_id

vc_discuss

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
discuss_id	**INTEGER**	PK	NN	UNSIGNED			AI
parent_id	INTEGER			UNSIGNED			
user_id	INTEGER			UNSIGNED			
tutorial_id	INTEGER			UNSIGNED			
sw_id	INTEGER			UNSIGNED			
news_id	INTEGER			UNSIGNED			
title	VARCHAR(256)						
body	VARCHAR(4096)						
create_datetime	DATETIME						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	discuss_id
vc_discuss_FKIndex1	Index	tutorial_id
vc_discuss_FKIndex2	Index	news_id
vc_discuss_FKIndex3	Index	parent_id
vc_discuss_FKIndex4	Index	user_id
vc_discuss_FKIndex5	Index	sw_id

vc_discuss

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
discuss_id	**INTEGER**	PK	NN	UNSIGNED			AI
parent_id	INTEGER			UNSIGNED			
user_id	INTEGER			UNSIGNED			
tutorial_id	INTEGER			UNSIGNED			
sw_id	INTEGER			UNSIGNED			
news_id	INTEGER			UNSIGNED			
title	VARCHAR(256)						
body	VARCHAR(4096)						
create_datetime	DATETIME						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	discuss_id
vc_discuss_FKIndex1	Index	tutorial_id
vc_discuss_FKIndex2	Index	news_id
vc_discuss_FKIndex3	Index	parent_id
vc_discuss_FKIndex4	Index	user_id
vc_discuss_FKIndex5	Index	sw_id

vc_branch

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
branch_id	**INTEGER**	PK	NN	UNSIGNED			AI
parent_id	INTEGER			UNSIGNED			
name	VARCHAR(45)		NN				
description	VARCHAR(255)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	branch_id
vc_branch_FKIndex1	Index	parent_id

vc_user

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
user_id	**INTEGER**	PK	NN	UNSIGNED			AI
role_id	INTEGER		NN	UNSIGNED			
firstname	VARCHAR(45)		NN				
surname	VARCHAR(45)		NN				
username	VARCHAR(45)		NN				
passwd	VARCHAR(80)		NN				
email	VARCHAR(60)		NN				
year_of_birth	INTEGER			UNSIGNED			
gender	VARCHAR(6)						
locality	VARCHAR(128)		NN				
skypename	VARCHAR(45)						
icq	VARCHAR(20)						
avatar_filepath	VARCHAR(60)						
suspended	BOOL		NN				
approved	BOOL		NN				
lastlogin	DATETIME						
create_datetime	DATETIME						
userprofile	BLOB						
usersettings	BLOB						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	user_id
vc_user_FKIndex1	Index	role_id

vc_user

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
user_id	**INTEGER**	PK	NN	UNSIGNED			AI
role_id	INTEGER		NN	UNSIGNED			
firstname	VARCHAR(45)		NN				
surname	VARCHAR(45)		NN				
username	VARCHAR(45)		NN				
passwd	VARCHAR(80)		NN				
email	VARCHAR(60)		NN				
year_of_birth	INTEGER			UNSIGNED			
gender	VARCHAR(6)						
locality	VARCHAR(128)		NN				
skypename	VARCHAR(45)						
icq	VARCHAR(20)						
avatar_filepath	VARCHAR(60)						
suspended	BOOL		NN				
approved	BOOL		NN				
lastlogin	DATETIME						
create_datetime	DATETIME						
userprofile	BLOB						
usersettings	BLOB						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	user_id
vc_user_FKIndex1	Index	role_id

Appendix B

Screenshots

Figure B.1: *Login* page

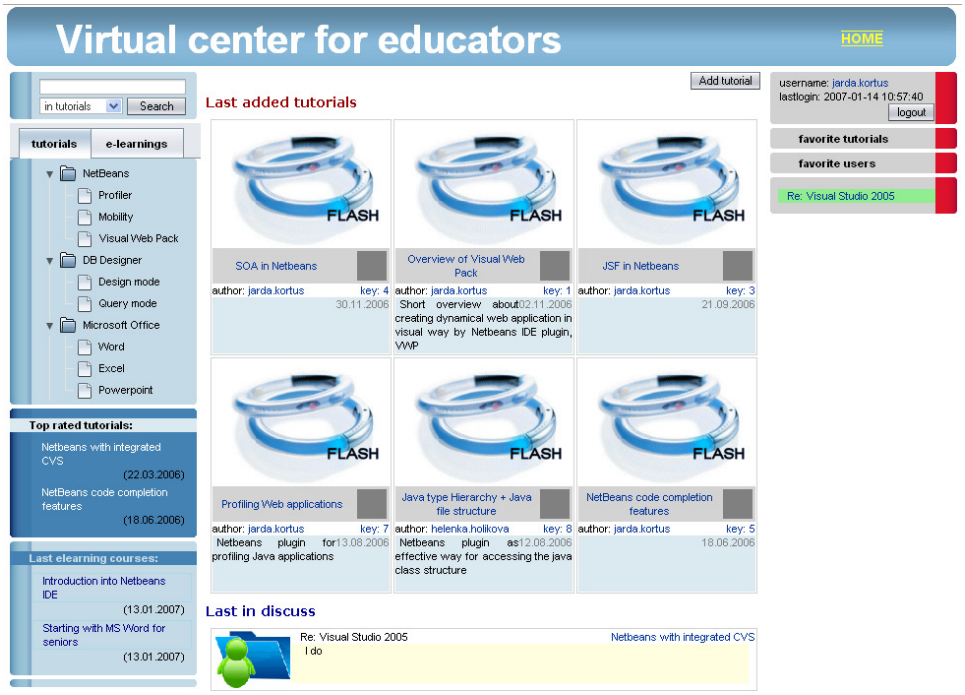


Figure B.2: *Hello* page

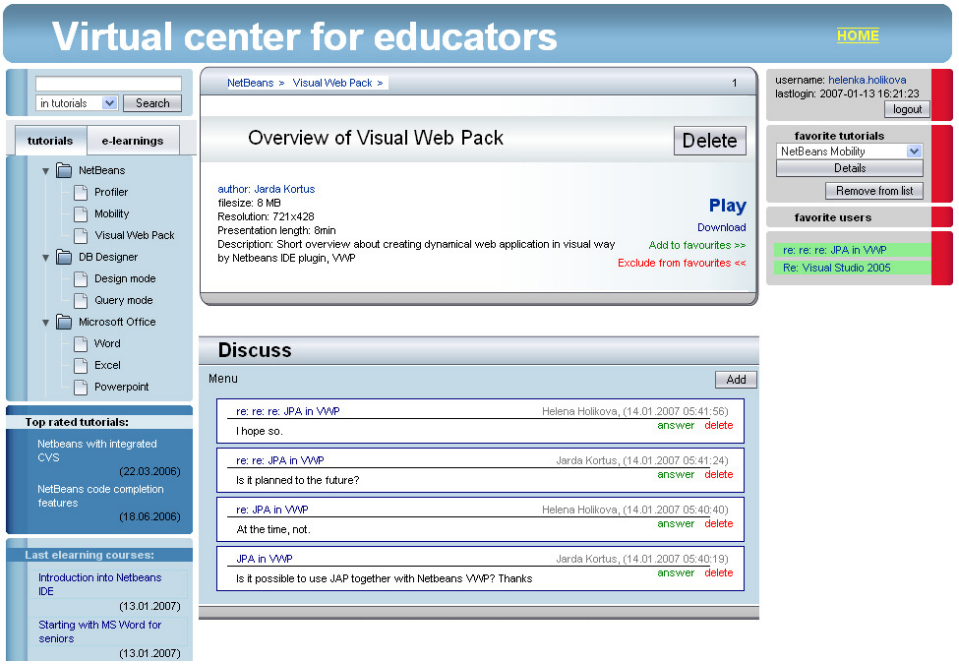


Figure B.3: *Tutorial* page

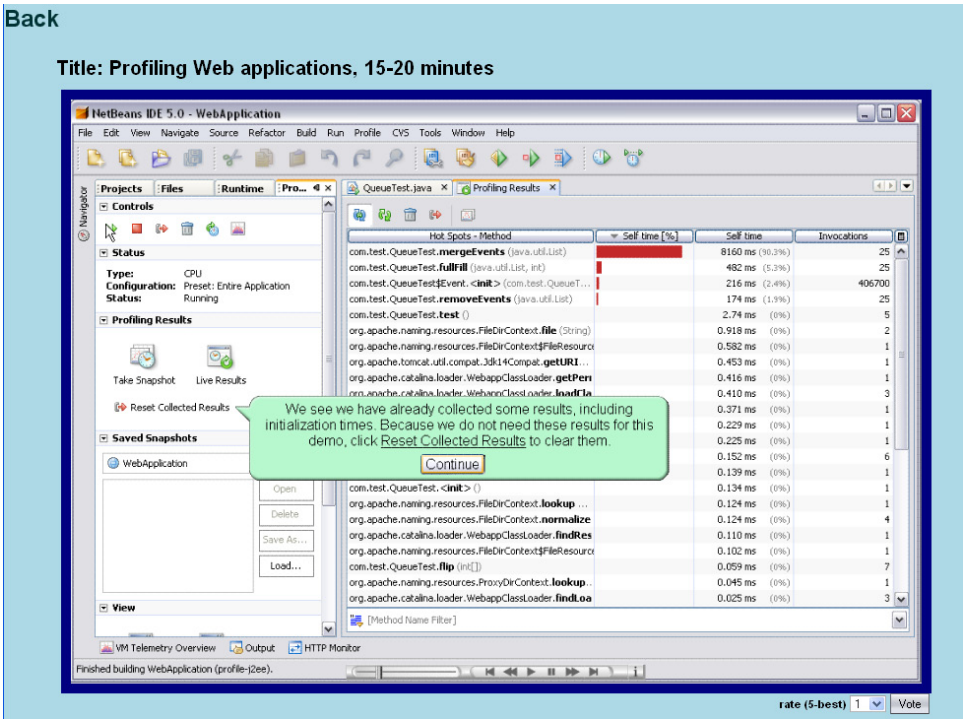

Back
Title: Profiling Web applications, 15-20 minutes
NetBeans IDE 5.0 - WebApplication
We see we have already collected some results, including initialization times. Because we do not need these results for this demo, click Reset Collected Results to clear them.
Continue
rate (5-best)
Vote

Figure B.4: *Play* page

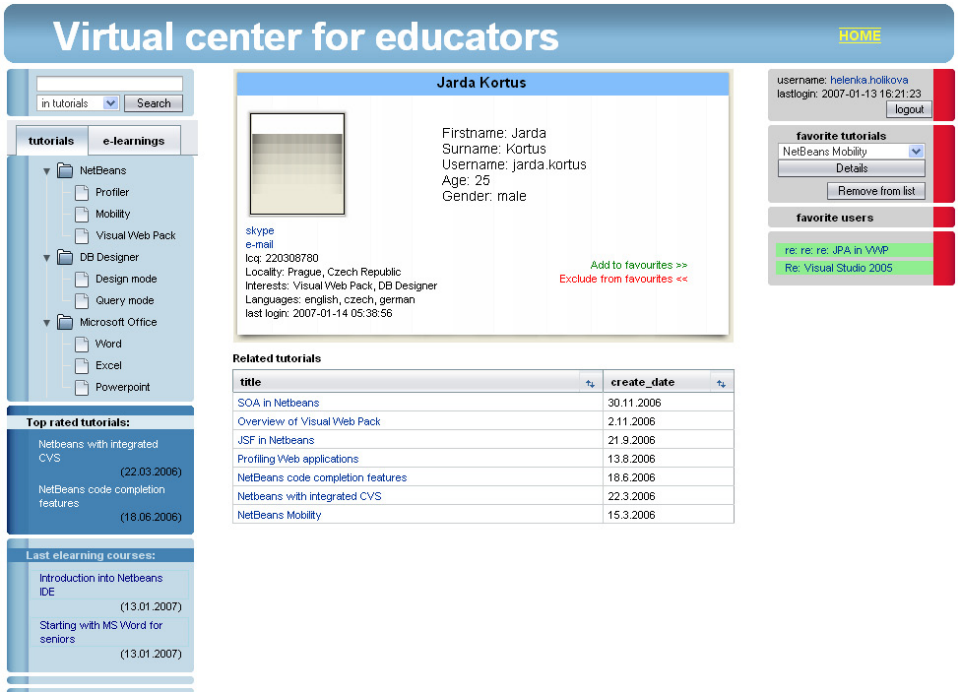


Figure B.5: *User* page

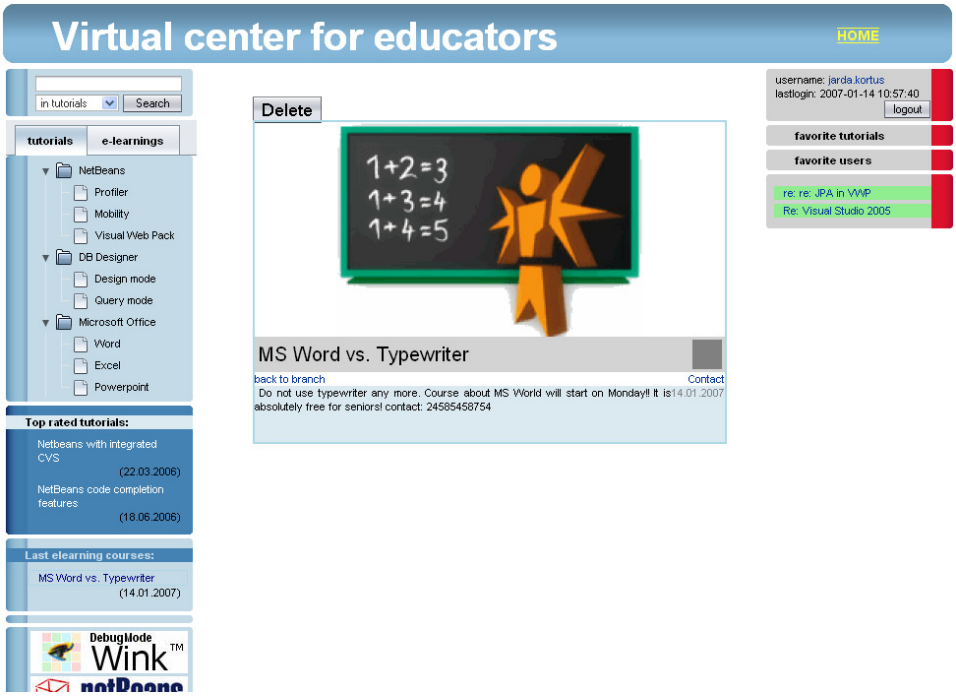


Figure B.6: *Elearn* page

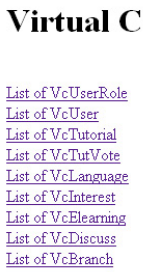
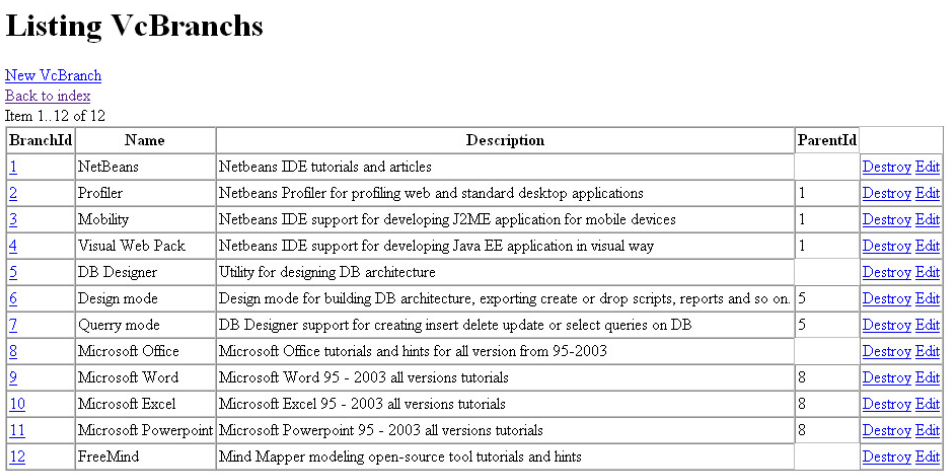


Figure B.7: *Index* page of vc_jpa application

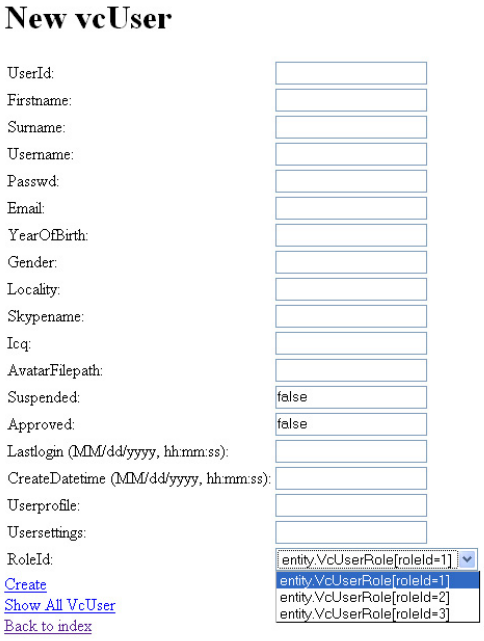


Listing VcBranchs

New VcBranch
Back to index
Item 1..12 of 12

BranchId	Name	Description	ParentId	
1	NetBeans	Netbeans IDE tutorials and articles		Destroy Edit
2	Profiler	Netbeans Profiler for profiling web and standard desktop applications	1	Destroy Edit
3	Mobility	Netbeans IDE support for developing J2ME application for mobile devices	1	Destroy Edit
4	Visual Web Pack	Netbeans IDE support for developing Java EE application in visual way	1	Destroy Edit
5	DB Designer	Utility for designing DB architecture		Destroy Edit
6	Design mode	Design mode for building DB architecture, exporting create or drop scripts, reports and so on.	5	Destroy Edit
7	Querry mode	DB Designer support for creating insert delete update or select queries on DB	5	Destroy Edit
8	Microsoft Office	Microsoft Office tutorials and hints for all version from 95-2003		Destroy Edit
9	Microsoft Word	Microsoft Word 95 - 2003 all versions tutorials	8	Destroy Edit
10	Microsoft Excel	Microsoft Excel 95 - 2003 all versions tutorials	8	Destroy Edit
11	Microsoft Powerpoint	Microsoft Powerpoint 95 - 2003 all versions tutorials	8	Destroy Edit
12	FreeMind	Mind Mapper modeling open-source tool tutorials and hints		Destroy Edit

Figure B.8: *List* page of vc_jpa application



New vcUser

UserId:
Firstname:
Surname:
Username:
Passwd:
Email:
YearOfBirth:
Gender:
Locality:
Skypename:
Icq:
AvatarFilepath:
Suspended: false
Approved: false
Lastlogin (MM/dd/yyyy, hh:mm:ss):
CreateDatetime (MM/dd/yyyy, hh:mm:ss):
Userprofile:
Usersettings:
RoleId: entity.VcUserRole[roleId=1]
entity.VcUserRole[roleId=1]
entity.VcUserRole[roleId=2]
entity.VcUserRole[roleId=3]
Create
Show All VcUser
Back to index

Figure B.9: *New* page of vc_jpa application

Appendix C

Content of CD

```
CDROM:.
|   demos.html                                  Presentation demos
|   index.html                                  Welcome project page
|   install.cz.html                             Installation manual (Czech language)
|   install.en.html                             Installation manual (English language)
|
+---app                                         directory where you can find useful applications
|       com-sun-rave-sampleapps-complibs-ajax.nbm       AJAX components module for Netbeans 5.5
|       Firefox Setup 2.0.exe                   Mozilla Firefox internet web browser
|       jdk-1_5_0_06-windows-i586-p.exe         JDK 1.5 for Windows XP (other versions on
|                                                       http://java.sun.com/downloads/index.html)
|       mysql-connector-java-3.1.12.zip         MYSQL JDBC connector library, version 3.1
|       mysql-essential-5.0.24a-win32.msi       MYSQL Server 5
|       mysql-gui-tools-5.0-r3-win32.msi        MYSQL GUI tools for working with MySQL database in visual way
|       netbeans-visualweb-5_5-windows_0612.exe         Visual Web Pack for Netbeans 5.5
|       Opera_9.02_Eng_Setup.exe                Opera Web Browser 9
|       sjsas_pe-9_0_01-nb-5_5-win.exe          Sun Java System Application Server 9 + Netbeans 5.5
|       sjsas_pe-9_0_01-windows.exe             Sun Java System Application Server 9 (only)
|       wink20.exe                              Wink 2.0, application for creating Flash tutorials
|       ...
+---demos                                       
|   |   demos.html                              HTML file for accessing flash tutorials
|   |
|   \---src
|           070104_vc_firstpart.swf             Demo (Virtual Center - First part)
|           070104_vc_secondpart.swf            Demo (Virtual Center - Second part)
|           install1.swf                        Demo (Deploying and starting of the Virtual Center)
|           vc_JPA.swf                          Demo (Virtual Center - Table manager application, created by JPA)
|
+---repository                                  CVS repository folder
|
+---thesis                                      Masters thesis document
|   +---latex                                   LaTeX sources (inc. images)
|   |
|   +---other_sources                           Other sources (UML, DB scheme)
|   |
|   \---print
|           thesis.pdf                          Print version of the thesis
|
+---vc                                          Netbeans vc project folder
|   |
|   +---build                                   Netbeans Build directory
|   +---CVS
|   |
|   +---dist                                    Netbeans Dist directory
|   |   |   vc.war                              The Virtual Center Web ARchive file
|   |   |
|   |   \---javadoc                             JavaDoc
|   |
|   +---lib                                     folder with libraries
|   |
|   +---setup
|   |       jdbc_dataSource.sun-resource        JDBC data source configuration for the VC
|   |       mysqlPool.sun-resource              Connection Pool configuration for the VC
|   |
|   +---src                                     Source files
|   |
|   \---test                                    JUnit tests folder
|
\---vc_JPA                                      Netbeans vc_JPA project folder
```

Figure C.1: The list of enclosed CDROM

Bibliography

[1] Mary Campione – *The Java™ Tutorial,* Addison-Wesley, 2000

[2] Pavel Herout – *Učebnice jazyka Java™*, KOPP 2003, ISBN 80-7232-115-3

[3] Pavel Herout – *Java - Bohatství knihoven*, KOPP 2003, ISBN 80-7232-xxx-x

[4] Dave Ragget, Le Hors, Ian Jacobs – *HTML 4.01*
`http://www.w3.org/TR/html401/`

[5] Dušan Janovský – *Jak psát web* `http://www.jakpsatweb.cz/`

[6] *Web Developer* `http://www.webdeveloper.com/html/`

[7] B.Mannová, K.Vosátka – *Řízení softwarových projektů*
Nakladatelství ČVUT 2005, Praha ISBN 80-01-03297-3

[8] Danny Coward – *Java Servlet Specification*, Sun Microsystems 2006

[9] Group of authors – *XML Tutorial*, `http://www.w3schools.com/xml/default.asp`

[10] Jiří Kosek – *XML pro každého*, Grada Publishing 2000, ISBN 80-7169-860-1 `http://www.kosek.cz/xml/xmlprokazdeho.pdf`

[11] R.Orfali – *Essential Client/Server Survival Guide, Second Edition*,
John Wiley & Sons, 1997

[12] Java Community Process – *Java™ Servlet Specification Evaluation 2.5*,
`http://jcp.org/aboutJava/communityprocess/mrel/jsr154/index.html`

[13] M.Fowler *Patterns of Enterprise Application Architecture* Addison-Wesley, 2002

[14] J.Ball, D.Carson, I.Evans, S.Fordin, K.Haase, E.Jendrock – *The Java™ EE 5 Tutorial*,
`http://java.sun.com/javaee/5/docs/tutorial/doc/`
Sun Microsystems 2006, Update 1

[15] Sun Microsystems, Inc. – *JavaServer Pages 2.0*
`http://java.sun.com/products/jsp/download.html\#specs`

[16] Group of authors – *Wikipedia – Java Servlet* `http://en.wikipedia.org/wiki/Servlet` Web accessible document

[17] J.Hunter, W.Crawfor – *Java Servlet Programming*, O'Reilly, 1998

[18] Julie Leung, Jeremy Dawson – *Servlets* on Software Engineering Research Network,
`http://sern.ucalgary.ca/Courses/CPSC/547/W2000/webnotes/Servlets/`

[19] Sun Microsystems, Inc. – *Essentials of Filters*, http://java.sun.com/products/servlet/Filters.html/

[20] Internet Assigned Numbers Authority home page *MIME Media Types*, http://www.iana.org/assignments/media-types/

[21] Sun Microsystems, Inc. – *Java Servlet web site*, http://java.sun.com/products/servlet

[22] Sun Microsystems, Inc. – *The JavaServer Pages web site* http://java.sun.com/products/jsp

[23] Sun Microsystems, Inc – *JavaServer Pages Standard Tag Library 1.1 Tag Reference* http://java.sun.com/products/jsp/jstl/1.1/docs/tlddocs/

[24] Oracle – *About the Model 2 Versus Model 1 Architecture* http://www.oracle.com/webapps/online-help/jdeveloper/

[25] Hans Bergsten – *JavaServer Faces – Building Web-Based User Interfaces*, O'Reilly 2004

[26] *NetBeans home site*, http://www.netbeans.org/

[27] *Unified Modeling Language home site* http://www.uml.org/

[28] J.Arlow, I.Neustadt – *UML a unifikovaný proces vývoje aplikací* CP Books 2005, Brno ISBN 80-7226-947-X

[29] Sun Microsystems, Inc., – *Singapore 2006 Tech Days*, http://developers.sun.com/events/techdays/presentations/singapore2006/

[30] NetBeans team – *Java Persistence in the Java EE 5 Platform* http://www.netbeans.org/kb/55/persistence.html

[31] Sun Microsystems, Inc. – *Sun Java System Application Server Platform Edition 9 Administration Guide* http://docs.sun.com/app/docs/doc/819-3658

[32] Sun Microsystems, Inc. – *J2EE 1.4 Security* http://java.sun.com/j2ee/1.4/docs/tutorial/doc/Security3.html

[33] *Information on SSL specification* http://wp.netscape.com/eng/security

[34] Java Community Process – *Common Annotations for the Java™ Platform*

[35] Sun Microsystems, Inc. – *Application Deployment Guide for the Application Server* http://docs.sun.com/doc/819-3660

[36] Public comunity – *Top XML tutorials, blogs and forums* http://www.topxml.com/

[37] Frank Kelly – *My thoughts on best practices in software architecture and development as a whole* http://softarc.blogspot.com/